TAMPEREEN AMMATTIKORKEAKOULU
Tietotekniikan koulutusohjelma
Ohjelmistotekniikka

Tutkintotyö

Rami Rantanen

PELIYHTEISÖPALVELUN SUUNNITTELU JA TOTEUTUS

Työn valvoja Ohjelmistotekniikan lehtori Erkki Hietalahti

Tietotekniikka
Ohjelmistotekniikka
Rami Rantanen
Peliyhteisöpalvelun suunnittelu ja toteutus
Tutkintotyö 43 sivua
Työn ohjaaja Lehtori Erkki Hietalahti
Työn teettäjä Rami Rantanen
Huhtikuu 2007
Hakusanat peliyhteisöpalvelu, yhteisö

TIIVISTELMÄ

Tässä työssä on tarkoituksena rakentaa peliyhteisöpalvelu ja esittää sen suunnittelu- ja toteutusprosesseihin käytetyt menetelmät. Tavoitteena on toteuttaa vakaa ja monipuolinen järjestelmä, jossa pyritään huomioimaan erityisesti tehokkuus, käytettävyys ja ylläpidollinen helppous. Oleellinen osa työstä on peliaulatila, jossa on mahdollista yhdistää useita eri pelejä samaan sovelmaan. Käyttäjät voivat pelata pelejä toisiaan vastaan, keskustella ja liittyä luotuihin peleihin katsojiksi.

Koko palvelu sisältää monia ominaisuuksia, joista tärkeimpiä esitetään yksityiskohtaisesti. Toteutuksiin käytetyistä lähdekoodeista annetaan esimerkkeinä tärkeimmät osat. Palvelu on toteutettu hyvin käyttäjäkeskeisesti ja monet toiminnot mukautuvat dynaamisesti sivustolle kirjautuneiden käyttäjien omien asetusten perusteella.

Aluksi tässä työssä kuvataan yleiset ohjelmistokehityksen menetelmät, joita palvelun rakentamisessa käytetään. Tämän jälkeen oleellisena osana on peliaulatilan suunnittelu ja toteutus, mikä sisältää asiakas- ja palvelinjärjestelmät ja niiden välisen kommunikointiprotokollan. Dokumentin lopussa kuvataan palvelun sisältämät muut ominaisuudet.

Computer Engineering
Software Engineering
Rami Rantanen Designing and producing online game community
Engineering Thesis 43 pages
Thesis Supervisor Lecturer Erkki Hietalahti
Commission Rami Rantanen
April 2007
Keywords game community, community

ABSTRACT

In this thesis, the main purpose is to construct a game community service and describe the methods that are used in design and production processes. Objective is to produce a stable and versatile system and there is a tend to observe especially efficiency, usability, and easy administration. The essential part is the game lobby where it is possible to include many different games to the same applet. Users can play games with each other, communicate, and join to games as a spectators.

The whole service includes many features and the most important of them will be presented more specifically. The service has been produced with a very user centric view and many controls can be adjusted dynamically depending user's personal settings.

First, this thesis will describe the common software engineering methods that are used to produce the service. After that, the essential part is to desing and construct the game lobby which includes server and client systems and their communication protocol. At the end of this document, there will be explained other features.

Tietotekniikka
Ohjelmistotekniikka
Rami Rantanen

ALKUSANAT

Tämä työ on tehty vuoden 2006 ja kevään 2007 aikana. Alun perin harjoittelumielisestä harrasteesta lähtenyt kokeilu on kehittynyt monipuoliseksi palveluksi. Sivuston käyttäjämäärä on nopeasti lisääntynyt ja palvelu on saanut paljon positiivista palautetta.

Tällä hetkellä palvelu on toiminnassa ja se löytyy Internet-osoitteesta http://delfia.org. Mielenkiintoa palvelun kehittämiseen ovat pitäneet yllä jatkuva uuden oppiminen ja erityisesti useat aktiivisesti mukana olleet käyttäjät.

Tampereella 26. huhtikuuta 2007

Rami Rantanen

KÄYTETYT TERMIT JA LYHENTEET

Java	Laitteistosta riippumaton oliopohjainen ohjelmointikieli.
PHP	Erityisesti dynaamisiin Internet-sivuihin käytetty kieli.
Gentoo	Eräs Linuxin monista levitysversioista.
Peliaula	Internet-selaimessa käytettävä monen samanaikaisen käyttäjän tila, johon on integroitu useita eri sovelluksia.
Aulapalvelin	Palvelinlaitteistossa käytettävä sovellus, joka huolehtii peliaulajärjestelmän pääaulan toiminnasta.
Pelipalvelin	Palvelinlaitteiston sovellus, joka huolehtii peliaulaan integroidun yksittäisen pelin toiminnoista.
MySQL	Avoimen lähdekoodin relaatiotietokantajärjestelmä.
MyISAM	MySQL:n oletustyyppi tauluille. Tekniikkana nopea, mutta ei tue transaktioita.
InnoDB	Tietokantataulujen tyyppi, joka tukee transaktioita.
TCP	Transmission Control Protocol. Yhteyksien muodostamiseen käytettävä tietoliikenneprotokolla.
Porttikielto	Yksittäisiä käyttäjiä voidaan estää käyttämästä palvelun eri osia tietyksi ajaksi.

SISÄLLYSLUETTELO

1 JOHDANTO

Tämän tutkintotyön tarkoituksena on suunnitella ja toteuttaa peliyhteisöpalvelu. Työ koostuu useasta toisiinsa läheisesti liittyvästä osasta, jotka yhdessä muodostavat yhteisöpalvelukokonaisuuden. Tavoitteena on saavuttaa tehokas ja monipuolinen järjestelmä, joka toimii vakaasti suuressakin kuormituksessa. Internetin käyttö on lisääntynyt nopeasti viime vuosina ja erilaiset yhteisöt ovat jatkuvasti voimakkaassa kehityksessä. Tämän työn yhteisöpalvelu sisältää oleellisena osana Java-ohjelmointikielellä toteutetun aulatilan, jossa voidaan hyödyntää useiden käyttäjien samanaikaista paikallaoloa. Aulaan voi yhdistää eri sovelmia ja tällä hetkellä se sisältää useita eri pelejä, jolloin paikalla olevat käyttäjät voivat keskustella ja samalla pelata pelejä toistensa kanssa. Erityisinä ominaisuuksina järjestelmä mahdollistaa katsojien liittymisen peleihin ja yksinpelitulosten tallennuksen ennätyslistoille. Palveluun on myös integroitu automaattisesti toimiva turnausjärjestelmä.

Tässä työssä annetaan kuvaus palvelun toiminnasta erityisesti edellä mainittujen ominaisuuksien osalta. Koko palvelun rakentamiseen sisältyvät myös sekä XHTML- ja CSS-suunnittelu että keskustelualueiden ja monien muiden ominaisuuksien toteuttaminen, mutta niitä ei tässä työssä syvällisesti käsitellä. Tässä keskitytään Java-, PHP- ja MySQL-tekniikoihin, jotka liittyvät erityisesti palvelinsovellusten toteuttamiseen ja niiden kommunikointiin käyttäjien työasemien välillä. Sivuston ulkoasulliset osat jätetään pois, mutta suunnittelussa kuitenkin huomioidaan erityisesti käytettävyys ja selkeys.

Valmiita ratkaisuja vastaavanlaisen kokonaisen palvelun luomiseen ei ole saatavilla. Toteutetun peliaulan osalta Sun Microsystems on kehittämässä vapaasti saatavilla olevaa runkoa, josta on mahdollista rakentaa monen käyttäjän pelijärjestelmä. Kysyntää peliyhteisöpalveluille selvästi on, mutta Suomessa toiminta on hyvin vähäistä. Vain muutama yhtiö on rakentanut yhteisöpalveluita ja niiden suosio on suuri. Rahallisesti kannattavaksi virtuaaliyhteisöjen kehittämisen tekee Internet, joka mahdollistaa suuren asiakasjoukon saamisen palveluun nopeasti. Palvelun ollessa laadukas ja monipuolinen, on osa asiakkaista valmis maksamaan lisäominaisuuksista.

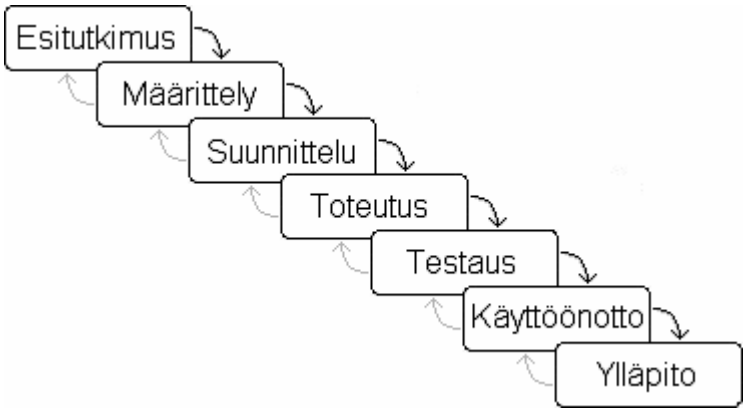
2 PALVELUN KEHITYSPROSESSI

Kokonaisuudessaan työ koostuu useasta pienestä, osittain rinnakkaisesta toteutuksesta. Tässä kappaleessa tutustutaan palvelun rakentamiseen yleisesti ohjelmistotuotannollisesta näkökulmasta. Tällöin huomioidaan toteutuksiin sovellettu tuotantomalli ja käydään läpi siihen kuuluvat vaiheet. Käytetystä mallista selviää sovellusten elinkaari kokonaisuudessaan.

2.1 Vesiputousmalli

Osatoteutuksiin sovellettiin ohjelmistokehityksessä yleisesti käytettyä kuvan 1 mukaista vesiputousmallia. Jokainen yksittäistoteutus muodosti selvän suoraviivaisesti etenevän oman prosessinsa, jossa tavoitteet ovat selviä, joten vesiputousmalli sopi näihin hyvin. Vesiputousmalli koostuu eri vaiheista, joissa esitutkimuksesta aloittaen jatketaan prosessin edetessä tasolta toiselle.

Jokaisessa vaiheessa tuotetaan jokin oleellinen vaihetuote, jota seuraavassa vaiheessa käytetään. Kaikkiin vaiheisiin liittyy tarkastuksia, testauksia ja muita laadun varmistavia toimenpiteitä. Niillä pyritään poistamaan kaikki virheet tuotettavasta sovelluksesta mahdollisimman aikaisessa vaiheessa /5/. Todennäköisesti mitä myöhemmässä vaiheessa kehitystyötä virheet löytää, sitä enemmän aikaa niiden korjaamiseen kuluu.



Kuva 1: Yksittäistoteutuksiin käytetty vesiputousmalli

2.2 Esitutkimus ja määrittely

Palvelun alkuperäisenä ideana ei ollut sen suosion lisääminen, vaan lähinnä eri tekniikoiden harjoitteleminen. Tavoitteena oli kartuttaa entistä monipuolisempaa osaamista, joka liittyy erityisesti Internet-teknologioihin ja Linux-palvelimen ylläpitoon. Palvelu ei ollut suunnattu alkuvaiheessa erityisesti kenellekään, vaan kaikki suunniteltiin ja toteutettiin vain oppimismielessä. Koska suunnittelut tehtiin vain itseä varten, olivat osatoteutusten tavoitteet aina selviä. Palvelun suosio on lisääntynyt hyvää vauhtia ja nykyisin sitä kehitetäänkin enemmänkin käyttäjien toiveiden mukaisesti. Palvelulla olisi tällä hetkellä hyvät edellytykset lisätä suosiotaan, mikäli niin haluttaisiin.

Käyttäjiltä tulee jatkuvasti ideoita palvelun kehittämiseksi, ja niistä jokaista pyritään jollakin tasolla tutkimaan ja kartoittamaan niiden toteuttamisen mahdollisuutta ja kannattavuutta. Kaikkia ehdotuksia ei resurssien puutteessa voida toteuttaa, mutta jotkin pääsevät esitutkimusvaiheessa muita pidemmälle, jolloin käyttäjät esittävät ideoidensa lisäksi samalla omia vaatimuksiaan. Esitutkimuksen tarkoituksena onkin asettaa yleiset järjestelmän vaatimukset ja siis asiakasvaatimukset, jotka määrittelevät asiakkaan eli toisin sanoen palvelun käyttäjien tarpeet. Kun nämä on mahdollisten lisäselvitysten jälkeen varmasti ymmärretty ja kun kyseiselle palvelun osalle olisi varmasti tiedossa potentiaalisia käyttäjiä eikä toteuttamiselle näy esteitä, voidaan sitä alkaa vielä tarkemmin määritellä.

Määrittelyvaiheessa tarkastellaan käyttäjien vaatimuksia, joiden perusteella saadaan ohjelmiston vaatimukset. Tällöin tutkitaan toteutuksen tarvitsemat sekä syöte- että tulostiedot ja niiden mahdolliset vaikutukset palvelun muihin osiin. Määrittely sisältää siis ohjelmiston toiminnot, käyttöliittymän ja ominaisuudet. Kaikissa näissä huomioidaan suoritusteho, laadukkuus ja käytettävyys. Erityisesti edellä mainittujen asioiden ollessa kunnossa saavat käyttäjät palvelusta positiivisen kuvan ja suurella todennäköisyydellä käyttävät palvelua vielä tulevaisuudessakin. Kaiken kaikkiaan sivusto pelejä myöten pyrkii olemaan käytettävyydeltään yhtenäinen, jolloin käyttäjien on helppo navigoida palvelun eri osissa.

2.3 Suunnittelu ja toteutus

Määrittelyvaiheen jälkeen suunnitellaan ohjelmassa tarvittavia algoritmeja ja tietorakenteita. Suunnittelun tarkoituksena on muuttaa edellisen vaiheen määrittely entistä teknisemmäksi kuvaukseksi, josta selviävät järjestelmän toteutustavat. Usein koko suunnittelu- ja toteutusvaihe on jaettu pieniin osiin, moduuleihin. Nämä osat ovat toisistaan riippumattomia ja selkeyttävät sekä suunnittelua että toteutusta. Ensimmäisenä suunniteltavana on koko sovelluksen rakenne eli arkkitehtuuri ja sen osamoduulien rajapinnat. Arkkitehtuuri jakautuu useisiin eri moduuleihin, joiden sisäinen rakenne suunnitellaan erikseen.

Hyvässä suunnittelussa yksittäisten moduulien sisäiset muutokset eivät vaikuta ulkopuolelle. Tällöin muutosten tekeminen, ylläpito ja uudelleen käytettävyys on entistä helpompaa, koska moduulit ovat toisistaan riippumattomia. Perusteellisesti tehtyjen valmisteluvaiheiden ja suunnittelun jälkeen aloitetaan varsinainen ohjelmointi. Ohjelmointikieliksi palvelussa on valittu Java ja PHP ja toteuttaminen suoritetaan pääasiassa GNU Nano -tekstieditorilla. Toteutusvaihe jatkuu, kunnes halutut ominaisuudet ovat toteutettuina ja ohjelma on kääntynyt virheettömästi.

2.4 Testauksen vaiheet

Testauksen tarkoituksena on löytää sovelluksesta virheet. Testausvaihe kokonaisuudessaan jakauu useaan eri osaan, ja ensimmäiset testaukset suoritetaan jo ohjelmoinnin yhteydessä. Tällöin tarkistetaan toteutettujen moduulien toiminnan oikeellisuus ja etsitään mahdolliset virhetilanteet. Myöhemmin ohjelman eri osat yhdistetään ja tarkistetaan rajapintojen toimivuus eri moduulien välillä. Tämän jälkeen ohjelmaa testataan pienellä testikäyttäjien ryhmällä, jotka pääsevät käyttämään uusia palvelun ominaisuuksia ennen niiden virallista julkaisua ja käyttöönottoa. Nämä järjestelmätestauksen toimenpiteet eivät vaikuta muun sivuston toimintaan ja toteutukset otetaan käyttöön vasta, kun kaikki toimivat varmasti halutuilla tavoilla eikä virhetilanteita ole löytynyt. Kaikkien näiden testausvaiheiden jälkeen julkaistaan kyseessä oleva ohjelma virallisesti kaikille käyttäjille.

2.5 Käyttöönotto ja ylläpito

Käyttöönottovaiheessa mahdolliset viimeiset löydetyt viat korjataan, järjestelmä muokataan täysin yhteensopivaksi uutta toteutusta varten, ja näiden toimenpiteiden valmistuttua toteutus julkaistaan virallisesti kaikille käyttäjille. Kaikki tässä työssä esitetyt toteutukset ovat päässeet vesiputousmallin viimeiseen vaiheeseen asti, paitsi turnausjärjestelmä, joka on kuitenkin jo kokonaisuudessaan teustauksen viimeisessä vaiheessa pienen käyttäjäryhmän käytettävissä. Mallin viimeinen vaihe on ylläpito, joka alkaa käyttöönotosta ja päättyy myöhemmin ohjelmiston poistamiseen kokonaan käytöstä.

Ylläpito sisältää tässä sekä laitteistoihin että ohjelmistoihin liittyvät toimenpiteet, joilla käyttäjien tyytyväisyys palvelussa varmistetaan. Tyytyväisyys muodostuu heille näkyvistä osista eli erityisesti ohjelmistojen virheettömyydestä, palvelun nopeudesta ja asiakaspalvelun laadusta. Näiden lisäksi ajan kuluessa myös tarpeet muuttuvat, jolloin käyttäjät odottavat uutta sisältöä koko palveluun ja myös päivityksiä jo olemassa oleviin sovelluksiin. Jo pienetkin päivitykset ja niistä tiedottaminen lisää käyttäjien mielenkiintoa palvelua kohtaan. Käyttäjille palvelimen laitteistoon liittyvät asiat näkyvät erityisesti nopeutena ja palvelun saatavuutena. Yleisesti järjestelmä on hyvin vakaassa tilassa sekä ohjelmistojen että laitteistojen osalta, joten palvelun saatavuus riippuukin enemmän Internet-yhteyden palveluntarjoajasta kuin palvelun mahdollisista muista virhetilanteista.

Käyttöön otettujen sovellusten jatkokehitykseen kuuluvat niiden korjaaminen, muokkaaminen ja muut asiat, jotka parantavat joko käytettävyyttä tai ylläpidettävyyttä. Kaksi yleistä ylläpidollista muutosta ovatkin virheiden korjaaminen ja ominaisuuksien lisääminen. Korjaustoimia tarvitaan, kun sovellus ei toimi oikein tai sen antama ulostulo ei ole vaatimusten mukainen. Korjaavan ylläpidon yksi ongelma on, että toimenpiteet saattavat aiheuttaa uusia virheitä, joita ei huomata. Täydellistävässä ylläpidossa lisätään jokin uusi ominaisuus sovellukseen, esimerkiksi vaatimusten muuttuessa ajan myötä /6/.

3 PELIAULAN RAKENTAMINEN

Peliaulan ideana on tuoda saman järjestelmän alle kaikki käyttäjät, jotka palvelua samaan aikaan ovat käyttämässä. Tällöin kyseiset käyttäjät näkevät toisensa ja voivat keskustella toistensa kanssa ja pelata joko yksityisesti tai julkisesti. Aulatilaan voi yhdistää eri sovelmia, joissa pystytään hyödyntämään samanaikaisten käyttäjien paikallaoloa.

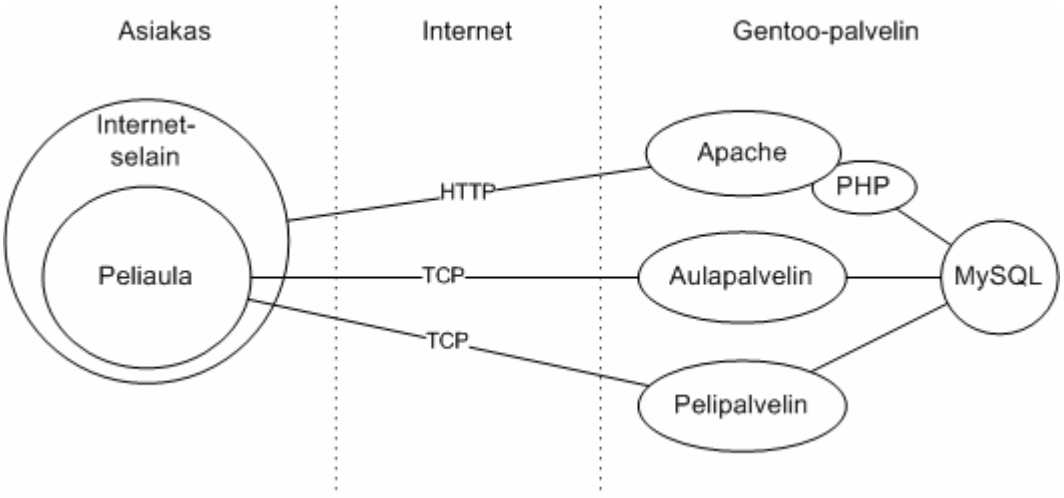
3.1 Järjestelmän arkkitehtuuri

Tällä hetkellä peliaula sisältää useita yksin- ja moninpelejä, joihin saa vastustajakseen kenet tahansa paikalla olevan käyttäjän tai vaihtoehtoisesti tietokonepelaajan. Järjestelmä mahdollistaa myös käyttäjien liittymisen peleihin vain katsojiksi. Sivustolla on mahdollisuus rekisteröidä nimimerkki itselleen, jolloin kukaan toinen käyttäjä ei voi samaa nimeä enää käyttää. Tämä mahdollistaa yksilöllisen kirjautumisen sivustolle, jolloin käyttäjä voi halutessaan määritellä itselleen sopivimmat asetukset. Asetukset tallennetaan MySQL-tietokantaan, jolloin sivusto muistaa automaattisesti nämä käyttäjittäiset asetukset, kun käyttäjä vierailee sivustolla seuraavan kerran.

Peliaulaan liittyminen edellyttää tällaisen rekisteröidyn nimimerkin käyttämistä väärinkäytösten ehkäisemiseksi. Käyttöehtoja rikkovien käyttäjien pääsy peliaulaan ja myös sivuston muihin osiin voidaan estää tietyksi ajaksi nimimerkin tai IP-osoitteen perusteella. Erikseen valvojiksi nimetyillä muutamilla käyttäjillä on oikeus käyttää peliaulaan integroitua hallintapaneelia, jonka kautta käyttäjille voi tarvittaessa asettaa porttikiellon. Kaikesta toiminnasta tallennetaan lokitietoja, jotta mahdolliset väärinkäytökset on mahdollista vielä myöhemminkin selvittää. Kokonaisuudessaan peliaulajärjestelmä koostuu useasta eri palvelinsovelluksesta. Pääpalvelin huolehtii aulatilan käyttäjälistasta, keskustelutoiminnoista ja käynnissä olevien pelien listauksista. Näiden lisäksi se tarkkailee alkavia turnauksia ja osaa automaattisesti luoda tarvittavat turnauspelit.

Pääpalvelimen lisäksi jokaisella eri pelillä on omat palvelinsovellukset, jotka huolehtivat kyseisten pelien tarvitsemista toiminnoista. Peliaulassa sisällä olevat käyttäjät ovat yhteydessä vain niihin palvelimiin, joita he kulloinkin tarvitsevat. Palvelinsovelluksilta ei edellytetä yhteyttä toisiinsa, joten ne voivat esimerkiksi sijaita lähiverkossa eri laitteistoissa toisistaan irrallaan. Pääpalvelinsovelluksella pitää olla pääsy sivuston käyttäjä- ja turnaustietokantoihin, kun taas pelipalvelimille riittää vain yhteys omaan tietokantaansa, jonne kyseisen pelin ennätystulokset tallennetaan.

Palvelu toteutettiin LAMP-tekniikkaa käyttävälle Gentoo-palvelimelle, jossa ovat käytössä Apache 2.0.58, MySQL 4.0.25 ja PHP 5.1.6. Oleellisin osa työstä eli peliaulan asiakassovelma ja palvelinpuolen sovellukset ohjelmoitiin Javalla käyttäen Blackdown JDK:n versiota 1.4.2. Javan API /1/ on erinomainen opas, josta selviävät kaikkien normaalien metodien käyttötavat. PHP:n uusimmat eli 5-sarjan versiot tukevat olio-ohjelmointia, joka mahdollistaa yksittäisten pienten ja uudelleen käytettävien osien rakentamisen, mikä merkitsee koko järjestelmälle entistä selkeämpää ohjelmointia ja ylläpitoa. Myös PHP:lle on olemassa kattava ohjesivusto /2/, josta eri funktioiden käyttötarkoitukset selviävät. Puhtaasti Javalla toteutettuna palvelinsovellukset toimivat suoraan eri käyttöjärjestelmillä, mikä tarvittaessa mahdollistaa helposti uusien palvelimien lisäämisen kokonaisuuteen ilman ongelmia. Kuvassa 2 on esitetty yleinen rakenne asiakkaan ja palvelinlaitteiston välillä.

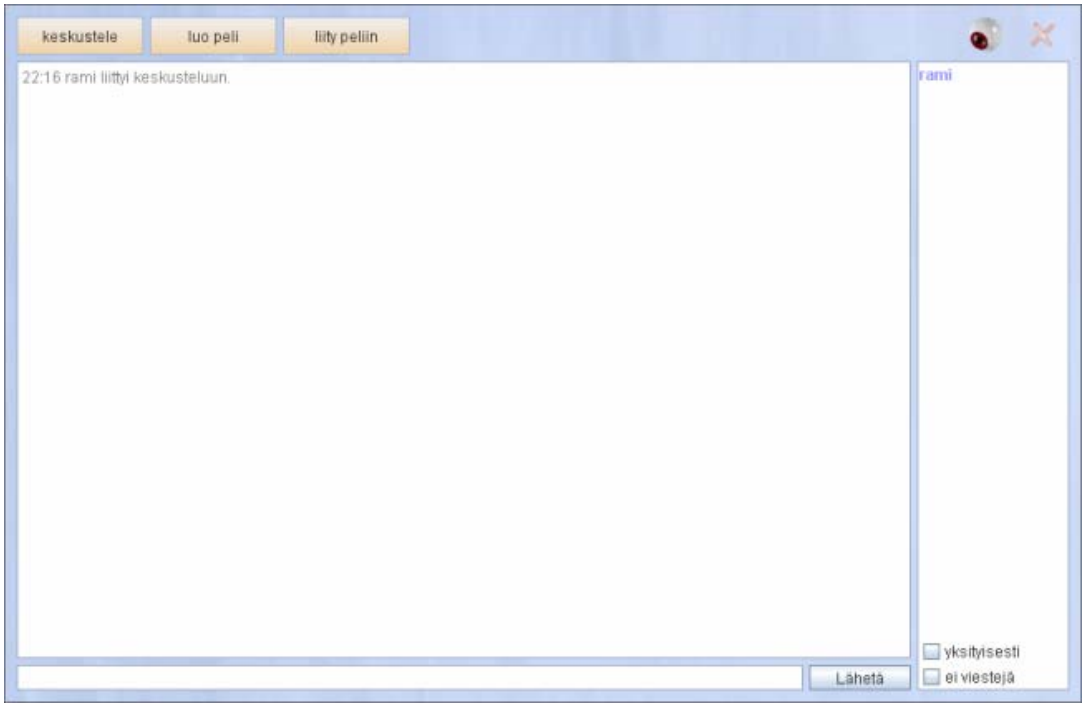


Kuva 2: Järjestelmän yleisen tason arkkitehtuuri

3.2 Asiakassovelma

Peliaulan toiminta näkyy käyttäjälle Internet-selaimella käytettävän Java-sovelman muodossa. Se tarkoittaa, että ainoa vaatimus sen toiminnalle on, että käyttäjien laitteissa pitää olla asennettuna Javan virtuaalimoottori. Aulatila jakautuu pääasiassa kolmelle eri näkymälle, joissa ensimmäisessä on keskusteluosio, toisessa luodaan pelit ja kolmannessa liitytään luotuihin peleihin joko katsojaksi tai pelaajaksi. Aulaan kirjauduttaessa aukeaa kuvan 3 kaltainen keskustelunäkymä.

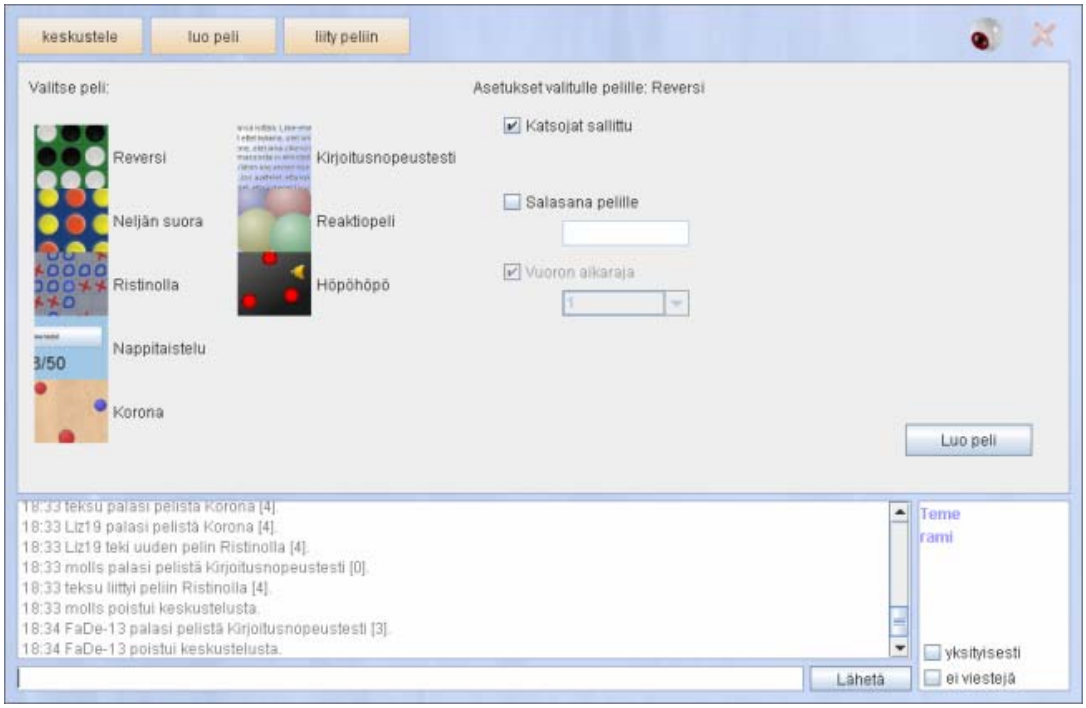
Sovelma on kokonaisuudessaan toteutettu käyttämällä Javan Swing-tekniikkaa. Siinä keskustelua varten on esillä suuri JTextPane-tekstialue ja paikalla olevien käyttäjien nimimerkit näkyvät listattuina oikeassa reunassa JList-komponentissa. Omat viestit käyttäjä kirjoittaa sovelman alareunan tekstikenttään. Käyttäjälistan yksittäisiä nimimerkkejä on mahdollisuus erikseen aktivoida, jolloin viestit voi lähettää yksityisesti vain valituille. Aulan eri näkymiä vaihdetaan yläreunan kolmella painikkeella.



Kuva 3: Keskustelunäkymä, jossa oikeaan reunaan listataan paikalla olevat käyttäjät

Kun käyttäjä haluaa luoda uuden pelin, aukeaa kuvan 4 kaltainen näkymä. Näkymän vaihdon jälkeen keskustelupaneeli ja käyttäjälista ovat edelleen näkyvissä pelin luontinäkymän lisäksi, mutta ne pienentyvät sovelman alaosaan. Tässä näkymässä ovat kaikki sovelmaan integroidut pelit listattuina ja niitä valitsemalla saa esiin peleittäiset asetukset, joita voi muokata haluamikseen. Esimerkiksi jokaiselle moninpelille on mahdollisuus valita, haluaako peliin hyväksyä katsojat ja vaatiiko peliin liittyminen salasanaa.

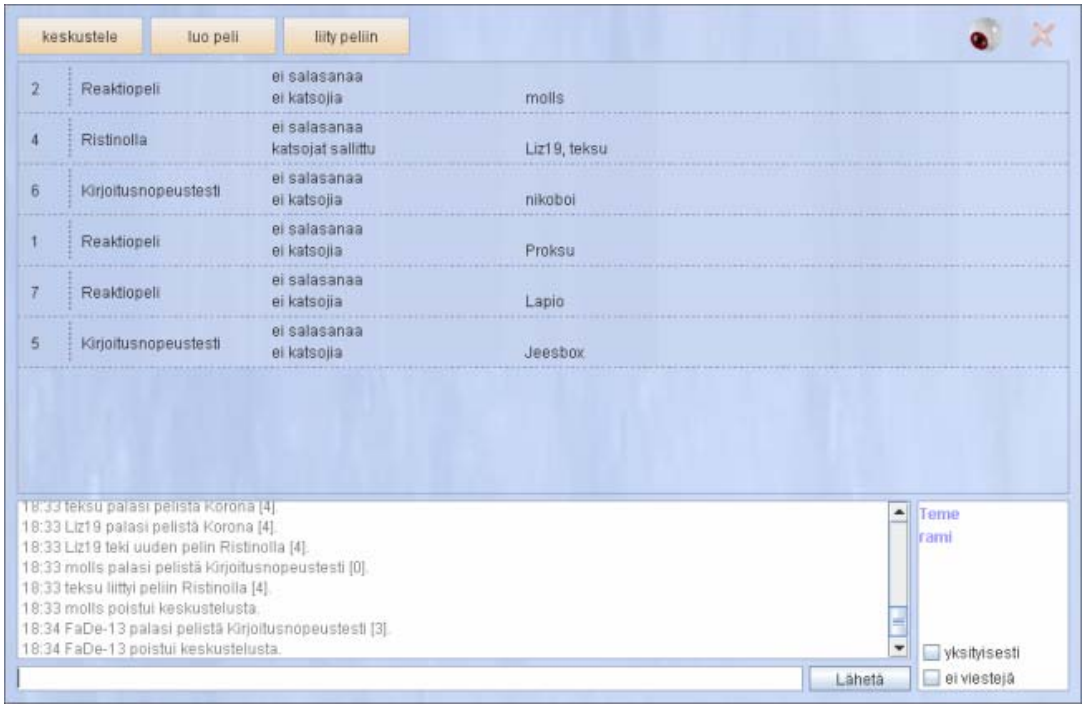
Asetusten valinnan jälkeen käyttäjä klikkaa ”Luo uusi peli”-painiketta. Tällöin kaikki halutun pelin tiedot ja asetukset syötetään uudelle pelioliolle, joka lähetetään pääpalvelimelle. Palvelin käsittelee olion valitsemalla sille ensimmäisen vapaan ID-tunnuksen. Uuden pelin luonut käyttäjä poistetaan käyttäjälistasta ja kyseinen peli lisätään pelilistaan. Näiden listojen muutokset lähetetään kaikille käyttäjille, paitsi pelin luoneelle käyttäjälle, jolle palvelin lähettää vain ID-tunnuksen. Vasta tämän tunnuksen vastaanotettuaan käyttäjän sovelma luo kyseisen pelin näkymän ja tällöin vastuu toiminnoista ja näkymän oikeellisuudesta siirtyy sovelmalta itse pelille.



Kuva 4: Uusien pelien luomiselle näkymä, jossa valitaan pelikohtaiset asetukset

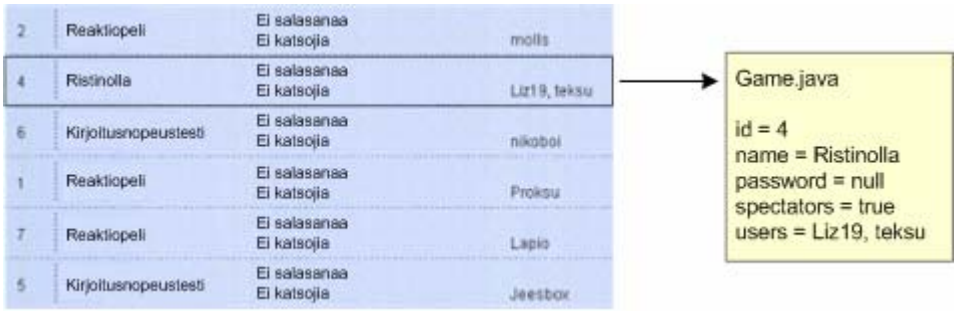
Viimeisestä näkymästä (kuva 5) liitytään jo aiemmin luotuihin peleihin. Käynnissä olevat pelit ovat listattuina JList-komponentissa ja niistä näytetään ID-tunnus, pelin nimi, pelissä sisällä olevat käyttäjät ja valitut asetukset. Pelaaja voi liittyä haluamaansa peliin klikkaamalla hiirellä kahdesti jotakin listassa näkyvää peliä. Mukaan liittyminen edellyttää, että pelin luonut käyttäjä on sallinut katsojat. Normaaleihin peleihin liityttäessä käyttäjä on aina ensin katsojatilassa, minkä jälkeen on mahdollista liittyä pelaajaksi, jos yksi pelin pelaajapaikoista on vapaana. Turnauspeleissä pelaajapaikat on arvottu valmiiksi, joten tällöin käyttäjä on automaattisesti pelaaja eikä katsoja.

Aina kun joku käyttäjä liittyy peliin, luo uuden pelin tai palaa takaisin pelistä, käsittelee palvelin tapahtumat ja lähettää kaikille aulatilassa oleville käyttäjille muuttuneiden pelien tiedot. Näillä tiedoilla käyttäjien sovelmat päivittävät pelilistan oikeanlaiseksi. Peli- ja käyttäjälistan muutoksista ilmoitetaan sovelman keskusteluosassa. JTextPane mahdollistaa eri värien käytön, joten käyttäjien kirjoittamat viestit, aulan yleiset tapahtumatiedot ja muut palvelimelta tulevat viestit ovat eri värillä selkeyden takia.



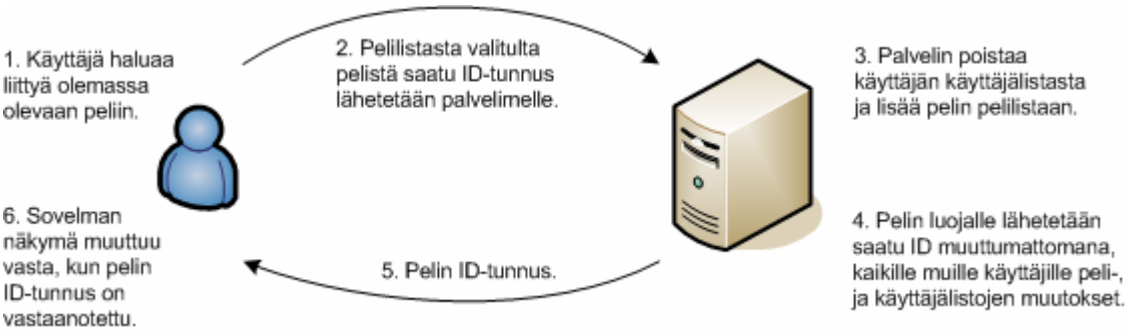
Kuva 5: Pelilistassa näkyvät kaikki käynnissä olevat pelit ja niiden oleelliset tiedot

Pelilista koostuu kuvan 6 mukaisesti peliolioista, joista jokainen sisältää yhdelle pelille tarvittavat tiedot. Lista käyttää javax.swing.JList-luokkaa, jossa jokainen solu sisältää yhden peliolion. Solut on toteutettu käyttämään erityistä renderöijää, johon peliolion tiedot sopivat. Jokainen solu on käytännössä javax.swing.JPanel-komponentti, johon peliolion tiedot sijoitetaan. Pelilista muuttuu aina, kun joku käyttäjä liittyy olemassa olevaan peliin, luo uuden pelin tai palaa pelistä takaisin aulatilaan. Näissä muutostilanteissa palvelin lähettää muuttuneet peliolioit käyttäjille, jotka saatujen tietojen perusteella päivittävät listansa oikeanlaisiksi.



Kuva 6: Pelilista koostuu peliolioista, jotka sisältävät kunkin pelin oleelliset tiedot

Kun käyttäjä haluaa liittyä peliin ja valitsee pelin listasta, lähetetään kyseinen ID palvelimelle, joka tämän jälkeen lisää käyttäjän nimimerkin kyseisen pelin tietoihin ja poistaa nimimerkin aulatilan yleisestä käyttäjälistasta. Sekä pelin luonnissa että peliin liittymisessä on samankaltainen prosessi (kuva 7), käyttäjät saavat lopulta molemmissa tapauksissa palvelimelta oikean pelin ID-tunnuksen. Tämän jälkeen sovelma osaa luoda oikeanlaisen pelin ja poistaa näkyvistä ylimääräiset paneelit.

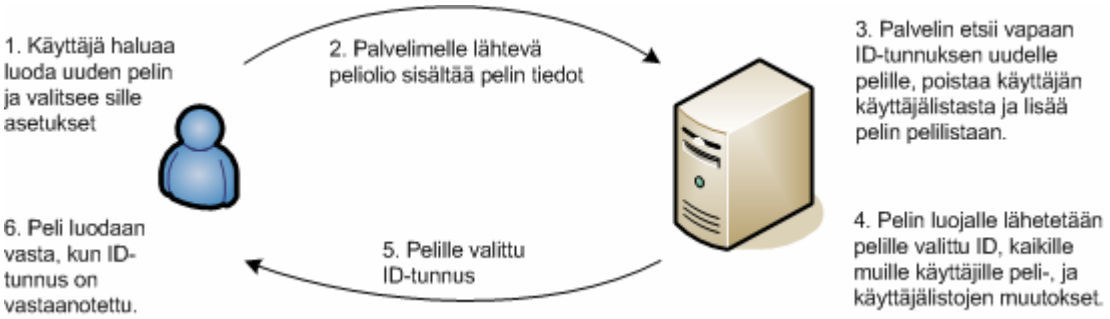


Kuva 7: Olemassa olevaan peliin liittymisessä suoritettava prosessi

3.3 Palvelinsovellukset

Käyttäjän liittyessä peliaulaan, muodostetaan yhteys pääpalvelimen ja kyseisen käyttäjän tietokoneen välille. Yhteydenmuodostuksesta ja viestinvälityksestä kerrotaan lisää jäljempänä. Yhteyden luonnin jälkeen käyttäjä lähettää ensimmäiseksi oman nimimerkkinsä, PHP:n sessiotunnuksensa ja IP-osoitteensa palvelimelle, joka näiden tietojen perusteella tarkistaa tietokannasta, onko kyseisellä käyttäjällä oikeus päästä peliaulaan. Jos käyttäjällä on porttikielto aulatilaan, lähettää palvelin käyttäjälle tiedot porttikiellon päättymisajankohdasta ja mahdollisen syyn, jos kiellon antaja on sellaisen kirjoittanut. Porttikieltoa ei tarkisteta erikseen joka pelissä, vaan se kohdistuu aina koko aulatilaan kerrallaan. Mikäli käyttäjää ei ole estetty, lähettää palvelinsovellus käyttäjälle peliaulassa sisällä olevien käyttäjien listan ja myös pelilistan, jos pelejä on käynnissä. Tällöin käyttäjä on sisällä aulatilassa ja näkee selaimessaan sovelman oikeassa reunassa listan paikalla olevista käyttäjistä.

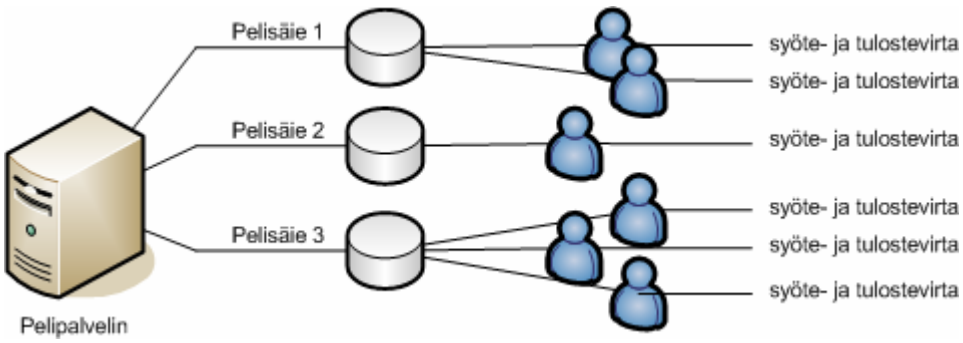
Sovelman eri välilehdiltä on mahdollista luoda uusi peli ja liittyä olemassa oleviin peleihin. Jokainen peli tarvitsee ID-tunnuksen, jonka perusteella pelit voidaan tunnistaa. Nämä tunnukset ovat myös näkyvissä aulatilan peliinliittymisnäkymässä jokaisen pelin kohdalla ja niiden perusteella toiset käyttäjät voivat niihin liittyä katsojaksi tai pelaajaksi. Sen vuoksi kuvan 8 mukaisesti pelin luontivaiheessa lähetetään ensin kaikki pelille valitut asetukset pääpalvelimelle Game-luokan ilmentymänä ja varsinainen peli luodaan sovelmassa vasta, kun käyttäjä on vastaanottanut pelille myönnetty ID-tunnuksen.



Kuva 8: Uuden pelin luomisprosessi

Palvelin vastaanottaa peliolion ja etsii sille ensimmäisen vapaan ID-tunnuksen, joka on mahdollisimman pieni kokonaisluku. Sen jälkeen peli lisätään aulatilan pelilistaan eli games-muuttujaan. Game-olio sisältää kaikki oleelliset tiedot yhdestä pelistä. Näiden tietojen perusteella muodostetaan sovelmassa näkyvä pelilista. Jokaiselle pelille on yksilöllinen ID-tunnus ja mikäli kyseessä on turnauspeli, on pelille annettu myös turnauksen ID, jonka perusteella pelit tunnistetaan. Lisäksi pelin tietoja ovat nimi, mahdollinen salasana, tieto onko katsojien pääsy peliin sallittu, käyttäjien nimimerkit ja muut pelikohtaiset asetukset. Sen jälkeen pelin luonut käyttäjä poistetaan aulatilan käyttäjälistasta. Pelille annettu ID-tunnus lähetetään kyseiselle käyttäjälle, ja muille käyttäjille lähetetään peli- ja käyttäjälistan muutokset. Uuden pelin luominen on tämän jälkeen pääpalvelimen osalta hoidettu.

Vasta kun käyttäjä vastaanottaa luodun pelin ID-tunnuksen, tehdään peli käyttäjän sovelmassa. Vastuu on tässä vaiheessa siirtynyt pääpalvelimelta kyseisen pelin omalle palvelinsovellukselle. Tällöin käyttäjälle aukeaa sovelmaan haluttu pelinäkymä ja yhteys pelipalvelimeen muodostetaan. Pelipalvelimet on toteutettu niin, että jokainen peliaulassa luotu peli on käynnissä omassa säikeessään. Yhteydenmuodostuksen jälkeen jokainen yksittäinen käyttäjäyhteys siirretään saadun pelin ID-tunnuksen perusteella oikeaan pelisäikeeseen. Tämä tarkoittaa sitä, että yksi pelipalvelin jakautuu moneen eri pelisäikeeseen ja näistä jokainen jakautuu edelleen jokaisen käyttäjän input- ja output- säikeisiin, jotka huolehtivat käyttäjän viestinvälityksestä (kuva 9).



Kuva 9: Jokainen pelipalvelin jakautuu useaan eri säikeeseen

Sekä pelin luonnissa että peliin liittymisessä käytetään pelipalvelimella samaa metodia, jolla käyttäjä asetetaan oikeaan pelisäikeeseen (listaus 1). Oikean pelisäikeen valinta tehdään pelin ID-tunnuksen perusteella. Mikäli ID löytyy pelilistasta, on peli jo käynnissä ja kyseessä on siis käyttäjän liittyminen mukaan olemassa olevaan peliin. Tuolloin vain lisätään käyttäjän input- ja output-virrat kyseiseen pelisäikeeseen. Jos ID-tunnusta ei löydy pelilistasta, on käyttäjä juuri luonut uuden pelin ja tällöin tehdään ensin uusi pelisäie, johon käyttäjän virrat lisätään.

```
public void setUserToGame(Integer gameID,
                          ObjectOutputStream out,
                          ObjectInputStream in)
{
    Game game = (Game)games.get(gameID);
    if(game == null)
    {
        game = new Game(this, gameID);
        game.addUser(out, in);
        games.put(gameID, game);
    }
    else
    {
        game.addUser(out, in);
    }
}
```

Listaus 1: Käyttäjä siirretään oikeaan pelisäikeeseen

Käyttäjän yhteys pääpalvelimeen pidetään auki myös silloin, kun käyttäjä liittyy peliin, jolloin toinen yhteys aukeaa johonkin pelipalvelimeen. Yksi syy kahden samanaikaisen yhteyden päälläoloon on se, että turnaustietokanta ja pääsy sinne on vain pääpalvelimella. Lisäksi kaikkien käyttäjien hallinnointi eli peliaulasta poistaminen ja porttikieltojen asettaminen onnistuu tällöin edelleenkin suoraan pääpalvelimelta peliaulan kautta, vaikka käyttäjät olisivatkin myös sisällä pelissä. Näitä toimenpiteitä kutsutaan valvojatoiminnoiksi, jotka ovat käytössä valvojiksi nimetyillä käyttäjillä. Nämä toiminnot vaikuttavat keskustelunäkymän lisäksi jokaiseen aulatilaan integroituun sovellukseen. Pääpalvelimen yhteys kuitenkin siirtyy entistä hiljaisempaan tilaan peliyhteyden auetessa. Tällöin pääyhtydessä siirretään vain tärkeimmät viestipaketit, joilla yhteyden voimassaolo varmistetaan ja lisäksi tarvittaessa edellä mainitun kaltaiset valvojaviestit.

3.4 Yhteyden muodostaminen

Yhteys asiakaslaitteen ja palvelimen välille muodostetaan käyttämällä java.net.Socket- ja java.net.ServerSocket-luokkia, joista ensimmäinen on käytössä asiakkaan laitteessa ja jälkimmäinen palvelimella. Palvelinsovellus on palvelimen taustalla aina ajossa oleva ohjelma, joka kuuntelee taukoamatta asetettuun porttiin tulevia yhteyksiä. Yhteyksien muodostaminen edellyttää, että palvelimen palomuurista on avattu käytettävät portit, jolloin ulkoverkosta on mahdollista muodostaa yhteys. Tietoturvauhkaa tästä ei kuitenkaan muodostu, varsinkin, koska sovellukset ovat toteutettu Java-kielellä. Tästä on se hyöty, että Java-sovellukset toimivat virtuaalimoottorin kautta, joka huolehtii esimerkiksi sovellusten muistin käytöstä. Sovellusten toiminta voi olla hieman puhtaasti laitteistoille natiivina käännettyjä ohjelmia hitaampaa, mutta tällä tavalla mitään muistin ylivuotovirheitä ei pääse vahingossakaan syntymään.

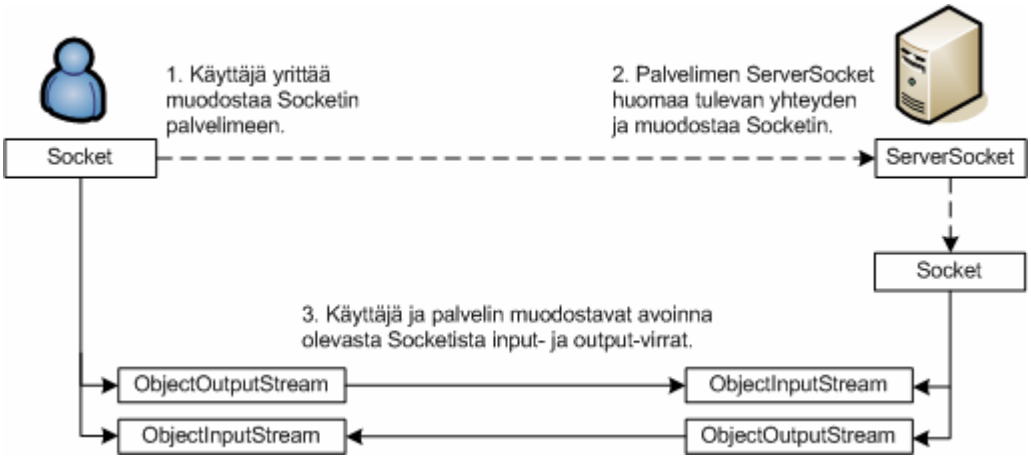
Kuuntelumetodissa (listaus 1) luodaan aluksi ServerSocket-luokasta olio, joka asetetaan käyttämään parametrina saatua porttia. Kun asiakas yrittää muodostaa yhteyttä samaan porttiin, muodostetaan ServerSocket-luokan accept()-metodilla uusi Socket-olio. Tämä tarkoittaa, että yhteys on muodostettu, jolloin tälle auki olevalle yhteydelle luodaan uusi säie, jossa yhteyttä valmistellaan. Yhteyden muodostaminen on kaksivaiheinen prosessi, jossa ensin yhteys valmistellaan, ja mikäli virheitä ei valmisteluvaiheessa tullut, siirretään käyttäjän yhteys oikeaan yhteyden käsittelijään. Valmisteluvaiheessa kyseinen käyttäjä ei vielä ole julkisesti muiden käyttäjien nähtävissä.

```
private void listen(int port) throws IOException
{
    ServerSocket serverSocket = new ServerSocket(port);

    while(true)
    {
        Socket socket = serverSocket.accept();
        new ConnectionInitializer(this, socket);
    }
}
```

Listaus 2: Palvelinsovelluksen metodi, jolla kuunnellaan tulevia yhteyksiä

Palvelinsovelluksessa yhteyttä valmistelevassa luokassa luodaan syöte- ja tulostevirrat, joiden jälkeen tehdään tarvittavat alustukset. Tällä hetkellä tähän kuuluu vain versiotarkistuksen tekeminen, jolla estetään epäyhteensopivien versioiden käyttämien. Sitä voi tapahtua sellaisissa tilanteissa, joissa käyttäjä on aikaisemmin ladannut järjestelmäänsä sovelman varhaisemman version ja uuden tullessa järjestelmä käyttää edelleen vanhaa versiota välimuististaan. Sekä asiakassovelmassa että palvelinsovelluksessa onkin tätä varten versionumerot ja valmisteluvaiheessa ne tarkistetaan. Tuolloin asiakas lähettää oman versionumeronsa palvelimelle ja mikäli versio on sama kuin palvelimella, hyväksytään yhteys, muuten lähetetään asiakkaalle ilmoitus versiovirheestä.

Yhteyden muodostamisessa kuvan 10 mukaisesti asiakas luo java.net.Socket-olion, jonka parametreinä ovat palvelimen osoite ja käytettävä portti. Jokaisella palvelinsovelluksella on käytössä eri portti. Palvelimella käynnissä oleva sovellus kuuntelee kulloinkin kyseessä olevaa porttia java.net.ServerSocket-luokan accept()-metodilla. Tästä muodostuu asiakkaan ja palvelimen välille yhteys, jonka välille luodaan tiedonsiirtovirta. Käytössä on oliovirta, jonka kautta voi suoraan siirtää Javalla luotuja olioita, kunhan ne pystyvät toteuttamaan Serializable-rajapinnan. Tämä mahdollistaa hyvin dynaamisen tiedonsiirtotekniikan, jolloin palvelimelle tai asiakkaalle voidaan siirtää suoraan halutun kaltainen olio.



Kuva 10: Syöte- ja tulostevirtojen muodostaminen käyttäjän ja palvelimen välille

Kirjautumisvaiheessa asiakassovelma muodostaa automaattisesti yhteyden pääpalvelinsovellukseen listauksessa 3 esitetyllä tavalla. Metodi on sama muodostettaessa yhteyttä pelipalvelimiin, mutta tällöin vain käytetty portti vaihtuu. Socketin muodostamisen jälkeen saadaan syöte- ja tulostevirrat, joista luodaan puskuroidut virrat ja näistä edelleen oliovirrat. Käytössä on siis pakkaamaton puskuroitu oliovirta, joka on riittävän nopea ja yksinkertainen ratkaisu tähän käyttöön. Toteutettuna on myös oliot pakkaava tekniikka, mutta näin pienillä siirrettävillä paketeilla se aiheuttaisi vain erittäin selvää viivettä pakettien siirtoihin. Yhteydet eivät ole salattuja eikä asiakkaan ja palvelimen välillä siirrettäviä viestejä suojata. Mikäli yhteys sattuisi jostakin syystä yllättäen katkeamaan, yritetään yhteys muodostaa automaattisesti uudestaan.

```
private void connect(int port, String host) throws IOException
{
    Socket socket = new Socket(host, port);

    InputStream is = socket.getInputStream();
    BufferedInputStream bis = new BufferedInputStream(is);
    ObjectInputStream in = new ObjectInputStream(bis);

    OutputStream os = socket.getOutputStream();
    BufferedOutputStream bos = new BufferedOutputStream(os);
    ObjectOutputStream out = new ObjectOutputStream(bos);

    out.flush();
}
```

Listaus 3: Asiakassovelman metodi, joka muodostaa yhteyden palvelinsovellukseen

3.5 Viestinvälitys

Kommunikointi palvelimen ja asiakaslaitteiden välillä tapahtuu siirtämällä olioita puskuroidussa virrassa. Jokainen olio sisältää siirrettävän viestin lisäksi tyypin, jonka perusteella tunnistetaan miten kyseinen viesti pitää käsitellä. Tyypit ovat määritetty erilliseen tiedostoon selkeästi nimetyiksi vakiokokonaisluvuiksi, joita voi käyttää jokaisesta viestejä käsittelevästä luokasta. Mahdollisuus siirtää olioita on erittäin dynaamista ja selkeää kaikkien jo toteutettujen ja myöhemmin tulevien järjestelmien kannalta Tällöin siirtoviesteihin voi sisällyttää juuri sellaista tietoa, mitä jokaisessa tilanteessa tarvitaan.

Vaikka olioiden siirtäminen on helppoa ja mahdollistaa dynaamiset järjestelmät, se edellyttää, että palvelinsovellukset ovat myös toteutettu Javalla. Tavuvirtoja tai puhtaasti tekstimuotoista protokollaa käyttämällä olisi mahdollista toteuttaa palvelinsovellukset matalan tason ohjelmointikielillä ja näin saada lisää tehokkuutta järjestelmään. Yleisesti yhteen tekstimuotoisen protokollan viestiin sisällytetään useita eri parametrejä, jotka palvelin erottelee toisistaan. Esimerkiksi ensimmäisenä parametrinä on viestin tyyppi, jonka perusteella loput parametrit osataan käsitellä. Javan olioita siirtämällä palvelin saa haettua viestistä tyyppitiedon ja kaikki parametrit niiden omilla metodeillaan. Merkkijonon käsittely on hitaampaa verrattuna tietojen saamiseen suoraan eri metodeilla. Siirtopaketit tekstimuotoisessa liikenteessä olisivat hieman pienempiä, mutta pakettien siirtoviive on tälläkin hetkellä niin pieni, ettei viivettä normaalitilanteessa voi havaita.

Listauksessa 9 on esitetty MSN-prokollan käyttämää tekstimuotoista viestinvälitystekniikkaa tilanteessa, jossa käyttäjä on kirjautumassa MSN-palveluun. Jokaisessa viestissä tyyppi selviää joka rivin ensimmäisestä parametristä, joka on aina kolmen merkin mittainen. Seuraava parametri on aina luku, jota kasvatetaan yhdellä jokaisen lähetetyn viestin yhteydessä. Palvelin lähettää asiakkaalle kuittauksen jokaiseen viestiin. Näissä lähetetyissä viesteissä parametrien määrä ja tietyissä tilanteissa myös niiden pituus on erittäin oleellista. Tämä ratkaisu rajoittaa siirrettävien viestien sisältöä eikä ole yhtä dynaaminen kuin olioviestien siirtäminen. Suuren tietomäärän purkaminen merkkijonosta voi olla hidasta, mutta matalan tason kielellä toteutettuna purkaminen olisi nopeampaa kuin Javan olioviestien käsittely.

```
Asiakas lähettää palvelimelle:
    VER 1 MSNP8 CVR0
Palvelin lähettää asiakkaalle kuittauksen:
    VER 1 MSNP CVR0
Asiakas lähettää palvelimelle:
    CVR 2 0x1035 win 5.1 i386 MSNMSGR 6.2.0208 MSMSGS a@b.c
Palvelin lähettää asiakkaalle kuittauksen:
    710 2
Asiakas lähettää palvelimelle:
    USR 3 TWN I a@b.c
Palvelin lähettää asiakkaalle kuittauksen:
    XFR 3 NS 207.46.110.41:1863 0 65.54.239.20:1863
```

Listaus 4: Esimerkki MSN-protokollan tekstimuotoisesta viestinvälityksestä

Olioiden siirtäminen tietovirrassa edellyttää, että kaikki siirrossa käytettävät oliot ovat sarjallistettuja. Kaikki viestioliot toteuttavat näin ollen java.io.Serializable-rajapinnan. Viestioliot sisältävät rakentajan, jossa tiedot asetetaan ja tämän lisäksi get-metodit kaikille tiedoille, joita kyseiseen olioon on mahdollista asettaa. Listauksessa 5 on esimerkki yksinkertaisesta toteutetusta viestioliosta, jonka tietoina ovat tyyppi ja merkkijono.

```
public class Message implements Serializable
{
    int type;
    String string;

    public Message(int type, String string)
    {
        this.type = type;
        this.string = string;
    }

    public void getType()
    {
        return type;
    }

    public String getString()
    {
        return string;
    }
}
```

Listaus 5: Sarjallistettu viestiolio sisältää tyypin ja muut tiedot, joita kyseisessä tapauksessa tarvitaan

Viestiolioissa käytetyt tyypit ovat vakiokokonaislukuja, jotka ovat määritettyinä Constants-rajapinnassa (listaus 6). Kun näitä tyyppitietoja tarvitaan, pitää kyseisen luokan määrittelyssä ottaa käyttöön rajapinta Constants. Tällä tavalla monessa eri luokassa voidaan käyttää samoja vakiomuuttujia. Muuttujat ovat Javan rajapinnoissa automaattisesti static final –määritettyjä, joten niitä ei tarvitse erikseen mainita.

```
public interface Constants {
    int USER_JOINED_GAME = 1;
    int USER_QUIT = 2;
    int USER_BACK_FROM_GAME = 3;
    int USER_CREATED_GAME = 4;
    int CHAT_MESSAGE = 5;
}
```

Listaus 6: Oleellisimmat yhteiskäytössä olevat vakiomuuttujat määritettynä rajapinnaksi

Kun yksi käyttäjä lähettää esimerkiksi keskustelussa viestin, lähettää hän silloin yhden viestiolion palvelimelle. Palvelimen yhteydenkäsittelijäsäie ottaa oliosta tyyppitiedon ja päättää sen perusteella viestin jatkotoimenpiteet. Käyttäjän poistuessa aulasta, kutsutaan metodia, joka huolehtii tarvittavista poistotoimenpiteistä. Listauksessa 7 on esimerkki yhteydenkäsittelijäsäikeestä. Olio luetaan Socketista luodusta virrasta readObject()-metodilla, joka muunnetaan viestioliomuotoon. Tämän jälkeen viestiolion metodeita voidaan kutsua suoraan, jolloin kaikista löytyvä getType()-metodi palauttaa viestin tyypin. Ehtolauseilla tarkistetaan viestistä saatu tyyppi, jolloin tiedetään millä tavalla kyseistä viestiä pitää käsitellä. Esimerkissä on jatkuva silmukkarakenne, jota suoritetaan niin kauan, että yhteydessä oleva käyttäjä lähettää poistumisviestin tai tapahtuu virhe viestin vastaanotossa. Näiden jälkeen kutsutaan poistometodia, jolla yhteys käyttäjän välille tuhotaan.

```
public void run()
{
    while(true)
    {
        Message message = null;
        int type = -1;

        try
        {
            message = (Message)in.readObject();
        }
        catch(IOException e)
        {
            e.printStackTrace();
            break;
        }

        if(message != null)
            type = message.getType();

        if(type == CHAT_MESSAGE)
        {
            ...
        }
        else if(type == USER_QUIT)
        {
            break;
        }
    }

    removeConnection();
}
```

Listaus 7: Palvelinsovelluksien yhteydenkäsittelijäsäie jokaiselle käyttäjälle

Mikäli viestiolio on sen tyypin perusteella tarkoitus lähettää kaikille yhteydessä oleville käyttäjille, ohjaa yhdestä käyttäjästä huolehtiva säie viestin pääluokan lähetysmetodille. Tämä metodi kutsuu jokaisen päällä olevan yhteyssäikeen omaa lähetysmetodia, joiden kautta viesti kulkee kaikille käyttäjille, jotka ovat vastaanottotilassa. Listauksessa 8 näkyy myös kuinka for-silmukka käy läpi kaikki yhteyslistan jäsenet järjestyksessä lopusta alkuun. Tämä on hieman nopeampaa kuin esimerkiksi yleisemmin käytetty muoto for(int i = 0; i < connections.size(); i++), jossa size()-metodia jouduttaisiin kutsumaan joka kierroksella.

```
for(int i = connections.size(); --i >= 0; )
{
    ((ConnectionHandler)connections.get(i)).send(message);
}
```

Listaus 8: Palvelinsovelluksen metodi, joka kutsuu kaikkien asiakasyhteyksien omia lähetysmetodeja

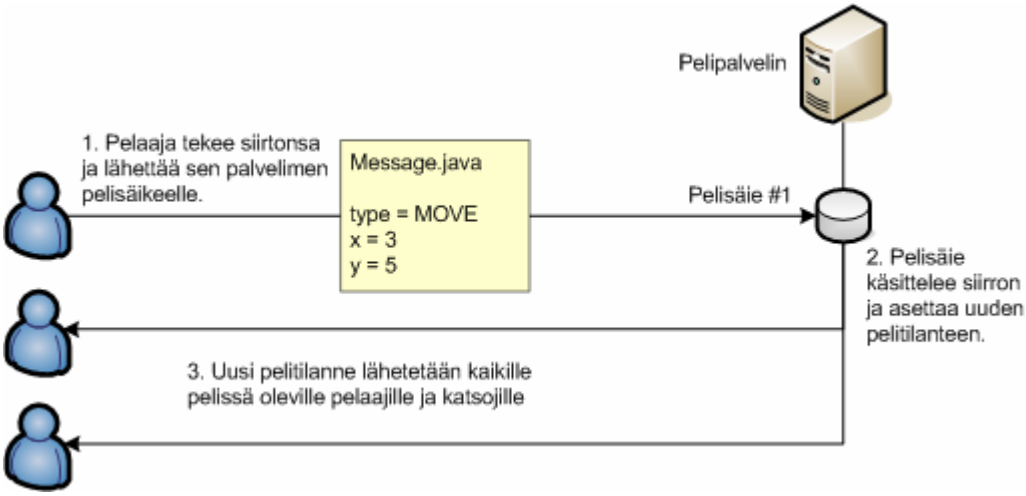
Hyöty on hyvin minimaalinen, mutta kuitenkin olemassa. Vastaavanlaisia keinoja lähdekoodin optimointiin ja sovellusten toiminnan tehostamiseen on käytetty yleisesti Java Optimization –oppaan /4/ mukaisesti. Lähdekoodit käännetään javac-ohjelman parametrillä –g:none, jonka ansiosta muodostettavat class-tiedostot vievät noin 20% vähemmän tilaa, joka tarkoittaa nopeampaa latausaikaa palvelun käyttäjille. Palvelin ja asiakas lähettävät viestinsä toisilleen listauksen 9 osoittamalla tavalla.

```
public synchronized void send(Message message) throws IOException
{
    out.writeObject(message);
    out.flush();
    out.reset();
}
```

Listaus 9: Palvelimella ja sovelmassa käytettävä metodi, jolla viestit lähetetään oliovirtaan

Lähettämisen jälkeen kutsutaan aina flush()- ja lopuksi reset()-metodeita. Muuten virta tallentaisi viimeksi lähetetyn viestin välimuistiin ja lähettäisi saman olion eteenpäin vielä seuraavillakin kerroilla. Vahvasti säikeistettyjen palvelinsovellusten johdosta viestien lähetysmetodit asiakkaille ovat synkronoituja, jotta vältytään virhetilanteilta. Erilaisia poikkeuksia voisi aiheutua, kun viestejä lähetetään liikaa samanaikaisesti, jolloin lähetysmetodi yrittäisi lähettää viestin jo käytössä olevaan virtaan.

Kaikkien käynnissä olevien pelien tiedot ovat palvelimella, jonka perusteella viestit yhteydessä oleville käyttäjille lähetetään. Koska pelit toimivat palvelimella, ei pääse syntymään tilanteita, joissa eri käyttäjillä olisi eri näkymä pelistä. Kun esimerkiksi pelaaja tekee siirtonsa, lähettää hän siirron tiedot peliä hoitavalle palvelinsäikeelle. Palvelin käsittelee viestin ja muokkaa pelitilannetta tarkoituksenmukaisella tavalla. Uusi pelitilanne lähetetään kaikille samaan pelisäikeeseen yhteydessä oleville käyttäjille kuvan 12 mukaisesti.



Kuva 11: Pelit ovat käynnissä palvelinsäikeissä, jotka huolehtivat pelitilanteiden oikeellisuudesta

Palvelimen ja asiakaslaitteiden välinen tiedonsiirto on pyritty minimoimaan, joten vain kriittisimmissä tapauksissa varmistetaan viestin perille meno vastapuolen vastausviestillä. Verkkoliikenteessä saavutetaan puskuroimalla huomattava etu puskuroimattomaan tiedonsiirtoon verrattuna. Mittausten mukaan nykyisellä palvelimen kokoonpanolla ja yhden megabitin Internet-yhteyden lähetyskaistallaan palvelinsovellus pystyy lähettämään ilman muuta suurempaa kuormistusta lähes 500 viestioliota sekunnissa. Yksi palvelimeen yhteydessä oleva käyttäjä pystyy näistä vastaanottamaan enimmillään yli 300 pakettia sekunnissa. Siirrettävät paketit ovat hyvin pienikokoisia ja siirtonopeus mahdollistaisi esimerkiksi reaaliaikaisemman pelin toteuttamisen. Verkkoliikenteessä ei käytetä olioiden pakkaamista, koska mittausten mukaan näin pienillä paketeilla siitä vain aiheutuisi enemmän viivettä.

Mikäli palvelimen kokonaiskuormitus nousee liian suureksi, on yksittäiset palvelinsovellukset mahdollista siirtää toisistaan irralleen eri laitteistoihin ilman ongelmia. Yhteyksien muodostaminen asiakkaan ja palvelimien välille edellyttää, että palvelimen palomuurista ovat tarvittavat portit auki. Tietoturvauhkaa tämä ei kuitenkaan muodosta, varsinkin kun palvelinsovellukset ovat toteutettuna Javalla. Esimerkiksi puskurin ylivuototilanteita ei pääse syntymään, koska Java huolehtii sisäisesti omasta muistinkäytöstään.

3.6 Tietokantayhteydet

Tietokantana järjestelmässä käytetään MySQL:ää, joka on GPL-lisenssin mukaisesti tähän tarkoitukseen maksuton ja dokumentaatio siihen on erittäin kattavaa /3/. Pääasiallisiksi tietokantataulujen tyypeiksi oli alunperin kaksi vaihtoehtoa, InnoDB ja MyISAM, joista verrattiin nopeutta ja ominaisuuksia toisiinsa. Nopeusmittaukset tehtiin käyttäen suurehkoa määrää erilaisia peruskyselylauseita, joita palvelussa tarvittaisiin. Tietokantoja käytetään palvelussa erityisesti käyttäjienhallinnassa ja käyttäjäkeskeisenä palveluna järjestelmän tietokanta sisältää jokaiselle rekisteröityneelle oleellisimpina tietoina nimimerkin, oikeustason ja ID-tunnuksen, johon monet muut tietokantataulut viittaavat. Tietokantoja käytetään myös esimerkiksi peliennätysten tallentamiseen ja tarvittavien turnauspelien luomiseen.

Käytössä olleella järjestelmällä INSERT-lauseiden suorittamiseen InnoDB-taululle kului aikaa noin 5,7 kertaa enemmän kuin MyISAM-taululle. SELECT-lauseiden suorittamisessa ero ei ollut aivan niin selvä, mutta myös niissä testeissä MyISAM oli 1,3 kertaa nopeampi. Näiden lisäksi testatut InnoDB-taulut veivät kiintolevytilaa jopa yli 36% vastaavan sisältöistä MyISAM-taulua enemmän. InnoDB mahdollistaa MyISAM-tauluihin verrattuna monia ominaisuuksia, esimerkiksi transaktioita, viiteavaimia ja viiteavainrajoitteita. Monet sovellukset eivät kuitenkaan tarvitse viite-eheyden valvontaa tietokannan hallintajärjestelmän puolelta, kuten ei tässäkään työssä toteutettu palvelu. Vaikka InnoDB tarjoaakin paremmat ominaisuudet, ei niille ole selvää tarvetta tässä järjestelmässä, joten taulujen tyypeiksi valittiin MySQL:n perustyyppi eli MyISAM.

Javassa MySQL-yhteyksiin käytetään java.sql-kirjastosta löytyviä Connection, DriverManager, ResultSet ja Statement –luokkia. Näiden lisäksi tarvitaan erikseen ladattava ajuritiedosto, joka tässä tapauksessa on mysql-connector-java-3.1.10-bin.jar. Listauksessa 10 on esitettynä metodi, jolla yhteys voidaan muodostaa. Parametreinä muodostuksessa annetaan tietokannan osoite, taulun nimi, käyttäjätunnus ja salasana. Myös käytetyn merkistön voi halutessaan antaa parametriksi ja tässä tapauksessa koko järjestelmä pyrkii käyttämään UTF-8:aa.

```
Connection connection = null;

public void connect() throws IOException
{
    Class.forName("com.mysql.jdbc.Driver");

    connection = DriverManager.getConnection(
        "jdbc:mysql://" +
        DB_HOST + "/" +
        DB_NAME +
        "?user=" + DB_USER +
        "&password=" + DB_PASS +
        "&useUnicode=true"+
        "&characterEncoding=UTF-8");
}
```

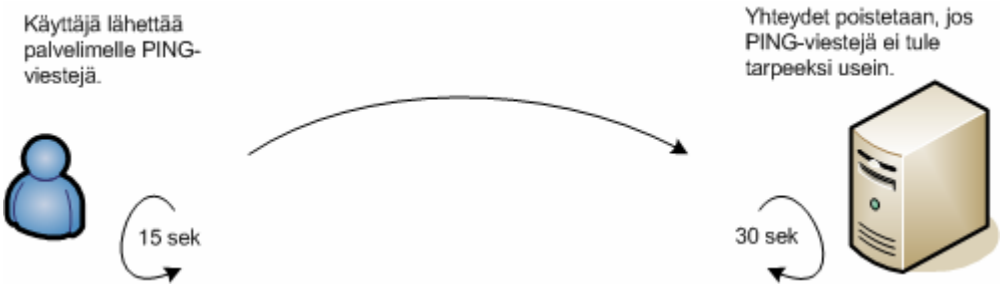
Listaus 10: MySQL-tietokantayhteyden luomiseen käytetty metodi

Connection-luokan createStatement()-metodilla saadaan Statement-olio, jonka kautta tietokantakyselyt suoritetaan. Kyseisellä oliolla on kaksi oleellista metodia: execute() ja executeQuery(). Näistä molemmat suorittavat annetun kyselyn, mutta vain jälkimmäinen palauttaa kyselyn tulokset ResultSet-oliona jatkokäsittelyä varten. Näin ollen execute() soveltuu esimerkiksi UPDATE ja INSERT –lauseiden käyttöön, kun taas executeQuery() paremmin SELECT-lauseille. ResultSet on tulosjoukko, joka sisältää riveinä kaikki kyselyn tulokset. Tulosjoukossa viitataan kerrallaan yhteen joukon riveistä. Käsiteltävää riviä vaihdetaan next()-metodilla, joka palauttaa false, kun uusia riviä ei enää tuloksissa ole. Statement- ja ResultSet-oliot on aina lopuksi suljettava, kun tarpeelliset kyselyt on suoritettu ja käsitelty. Tietokantakyselyt voivat aiheuttaa SQLException-poikkeuksen, jota varten ohjelmassa pitää aina olla toteutettuna poikkeuksenkäsittelijä. MySQL:n oppaassa on hyvin kattavat ohjeet kyselyille ja eri funktioiden käytölle.

3.7 Yhteyden voimassaolon tarkistus

Asiakkaan oman Internet-yhteyden ongelmatapauksissa voi käydä niin, että palvelinsovellus ei saa normaalimuotoista viestiä käyttäjän poistumisesta peliaulasta. Tämä tarkoittaa sitä, että käyttäjä näkyy peliaulassa muille käyttäjille, vaikka ei siellä todellisuudessa enää olisikaan. Tätä varten pitää tarkistaa säännöllisin väliajoin jokaisen yhteyden voimassaolo. Tarkistuksesta on huolehdittu niin, että jokainen yhteydessä oleva käyttäjä lähettää palvelimelle 15 sekunnin välein PING-tyyppisen viestin. Viestityyppinä PING on saman kaltainen kuin TCP/IP-protokollan ping-työkalu. Molemmissa idea on kokeilla halutun lautteen saatavuutta ja selvittää mahdollisia verkko-ongelmia.

Palvelimen tulisi saada näitä viestejä jokaiselta yhteydessä olevalta asiakkaalta 15 sekunnin välein. Palvelin tarkkailee saapuneiden viestien aikaleimoja ja mikäli yhtään viestiä ei puolen minuutin aikavälillä tule, poistetaan käyttäjän yhteys palvelimelta kuvan 12 mukaisesti. Tämä tarkistus on siis yksisuuntainen, joten palvelin ei vastaavia viestejä lähetä käyttäjille ollenkaan.



Kuva 12: Palvelin tarkkailee yhteyksiä säännöllisin väliajoin ja poistaa epäaktiiviset yhteydet

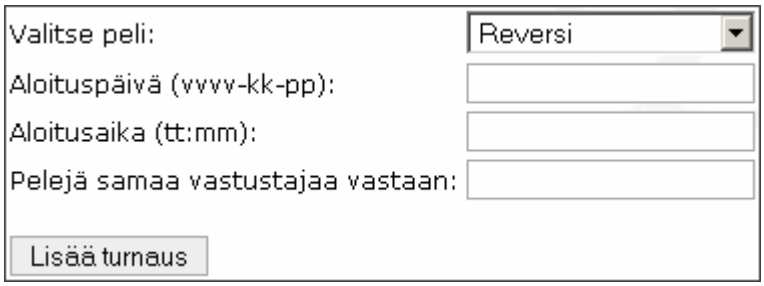
Yhteyksien varsinaiseen laatuun tai nopeuteen ei pääpalvelimessa oteta kantaa. Yhteyksien päällä pysymiseen riittää, että viestit siirtyvät järkevillä viiveillä. Peliaulaan integroiduilla muilla sovelluksilla on omat yhteydentarkastajansa, jotka tutkivat tarpeen mukaan tarkemmin yhteyden laatua. Esimerkiksi reaaliaikaisemmassa pelissä siirtoviive on tarkastettava paremmin, jotta liian huonon yhteyden omaavat pelaajat eivät häiritsisi muita pelaajia liikaa.

4 AUTOMAATTINEN TURNAUSJÄRJESTELMÄ

Turnausjärjestelmä on ominaisuus, joka lisää käyttäjien mielenkiintoa palvelua kohtaan selvästi. Se on ominaisuus, jota ei monista muista vastaavista palveluista edes löydy. Toteutettuna on muunnelma suorasta pudotustyylisestä turnaustyypistä, jossa järjestelmä automaattisesti hoitaa tarvittavien pelien luonnin ja koko turnauksen hallinnoinnin. Jäljempänä on tarkempi selitys järjestelmän toiminnasta.

4.1 Turnausten organisointi

Turnausjärjestelmän hallinnoinnissa ainoa osa, jossa käyttäjän toimia tarvitaan, on varsinainen turnauksen järjestäminen. Sivustolla on tätä varten paneeli (kuva 14), jossa tarvittavat oikeudet omaava käyttäjä voi määrittää järjestettäville turnauksille haluamansa asetukset. Tämän jälkeen kaikki toimii automaattisesti kyseisten turnausten osalta. Jokaiselle turnaukselle määritetään mikä peli on kyseessä, montako peliä samaa vastustajaa vastaan pelataan samalla kierroksella ja aloitusaika.

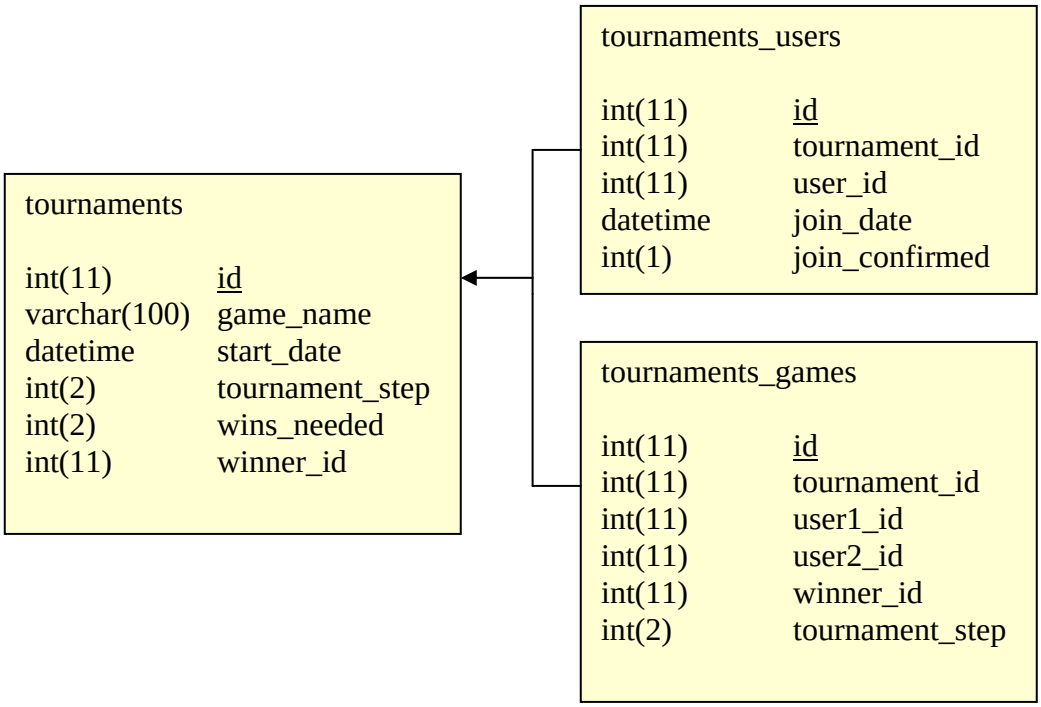


Kuva 13: Uuden turnauksen luomiseen käytettävä yksinkertainen lomake

Yhdellä kierroksella on mahdollisuus pelata useita pelejä samaa vastustajaa vastaan. Se pelaaja jatkaa seuraavalle kierrokselle, joka voittaa enemmän näistä kierroksen peleistä ja häviäjä putoaa turnauksesta pois. Kierroksittain pelaajien määrä puolittuu, jolloin lopulta jäljellä on vain yksi pelaaja. Turnauksen tietokanta soveltuu peleihin, joissa on aina vain 2 pelaajaa ottelua kohti. Näin ollen turnaukseen hyväksyttyjen pelaajien kokonaismäärä on aina oltava 2^X eli esimerkiksi 8, 16 tai 32. Turnauksen luonnin jälkeen se näkyy julkisessa turnauslistassa, jolloin jokainen rekisteröitynyt käyttäjä voi siihen ilmoittautua mukaan.

Turnaukseen osallistuminen pitää vahvistaa ennen turnauksen alkua ja tätä varten turnauslistaan ilmestyy vahvistuspainike 15 minuuttia ennen turnauksen aloitusta. Vahvistaneista pelaajista ensimmäiset pääsevät mukaan turnaukseen, jotka ovat hyväksytyn kokonaispelaajamäärän sisällä. Turnausjärjestelmän tietokanta sisältää kolme taulua kuvan 14 mukaisesti. Yksi sisältää sivuston hallintapaneelilla lisättyjen turnausten tiedot, toinen sisältää mukaan ilmoittautuneet käyttäjät ja kolmas kaikki pelatut pelit.

Uuden turnauksen luonnin yhteydessä syötetyt tiedot tallennetaan tournaments-tauluun. Turnaus saa tällöin ID-tunnuksen, johon muissa turnaustauluissa viitataan. Turnauksiin mukaan ilmoittautuneet pelaajat lisätään tauluun, josta heidät voidaan tournament_id-arvon perusteella myöhemmin yhdistää oikeaan turnaukseen. Jokainen pelattu peli lisätään pelitauluun, jossa alkutietoina ovat kahden pelaajan ID-tunnukset, turnauksen ID ja lisäksi luku, jolla peli voidaan yhdistää oikeaan turnauksen kierrokseen. Myöhemmin pelien tietoihin päivitetään voittajan ID. Viimeiseksi päivitetään turnauksen voittaja päätaulun tietoihin.



Kuva 14: Turnausjärjestelmän tietokantarakenne

4.2 Turnauspelien automaattinen luominen

Pääpalvelinsovellus huolehtii turnausjärjestelmästä ja ensimmäisenä tehtävänä on havaita alkavat turnaukset. Tätä varten pitää säännöllisin väliajoin suorittaa tietokantakysely, joka tarkistaa annettuista turnaustiedoista alkamisajan ja vertaa sitä sen hetkiseen aikaan. Tarkistus suoritetaan 15 minuutin välein ja turnaus aloitetaan, jos sen alkamisaika on ±10 minuutin sisällä sen hetkisestä oikeasta ajasta. Tietokannasta haetaan listauksen 11 mukaisesti turnauksen ID-tunnus, pelin nimi ja samaa vastustajaa kohti tarvittavien voittojen määrä. Turnaustietokannassa step-arvo kertoo missä tilanteessa kyseinen turnaus on menossa, 0 tarkoittaa ettei turnausta ole aloitettu vielä ja siitä eteenpäin sitä kasvatetaan yhdellä joka kierroksella.

```
SELECT id, game_name, wins_needed
FROM tournaments
WHERE step = 0
AND start_date < (NOW() + INTERVAL 10 minutes)
AND start_date > (NOW() - INTERVAL 10 minutes)
```

Listaus 11: SQL-kysely, jolla tarkistetaan onko turnauksia alkamassa

Mikäli alkava turnaus löytyi, tarkistetaan seuraavaksi siihen ilmoittautuneet käyttäjät. Jos ilmoittautumisensa vahvistaneita käyttäjiä on tarpeeksi, valitaan niistä vain ne ensimmäiset, jotka mahtuvat 2^X-ryhmään. Nämä mukaan hyväksytyt käyttäjät sijoitetaan satunnaiseen järjestykseen listaan. Lista käydään kokonaisuudessaan läpi for-silmukalla ja vierekkäisille paikoille arvotut käyttäjät ovat toistensa vastustajia. Näiden perusteella luodaan uudet pelit listauksen 12 mukaisesti.

```
for(int i = 0; i < users.size() - 1; i += 2)
{
    Game game = new Game();
    game.setName(name);
    game.setInfo("[turnaus]");
    game.addUser((User)users.get(i));
    game.addUser((User)users.get(i + 1));
    game.setSpectators(true);
    game.setTournamentID(id);

    server.addGame(game);
}
```

Listaus 12: Turnauspelien luominen

4.3 Tulosten tallennus ja turnauksen lopetus

Jokaisen kierroksen alussa kaikki luodut pelit lisätään tietokantaan. Kun yksittäinen peli pelattu, päivitetään sille voittajapelaajan tiedot. Tässä vaiheessa myös tarkistetaan aina onko saman kierroksen pelejä vielä pelaamatta. Mikäli pelejä ei ole käynnissä, tarkistetaan onko tarpeeksi pelaajia jäljellä seuraavaa kierrosta varten. Turnaus määritetään päättyneeksi siinä tilanteessa, kun kierroksella ei ole pelejä enää käynnissä ja vain yhdessä pelissä on voittajat selvillä. Tällöin koko turnauksen voittaja löytyy siitä pelistä, jossa on suurin mahdollinen kierrosarvo asetettuna. Turnauksen päättyessä lähetetään peliaulaan julkinen viesti kaikille käyttäjille turnauksen päättymisestä ja sen voittajasta.

Turnauksen kulkua voi seurata sivuston puolelta, jossa turnauksen kulusta piirretään PHP-skriptillä koko ajan kaaviota. Skripti päivittää kaaviota automaattisesti uusien kierrosten alkaessa ja turnauksen päättyessä, jolloin kaaviosta näkee selkeästi missä vaiheessa turnaus on. Kaavion ideana on piirtää jokaisella kierroksella pelatut pelit eri tasolle ja jokaisen pelin kohdalle pelaajien nimet. Tarvitut yksinkertaiset kuviot saadaan piirrettyä suoraan käyttämällä PHP:n GD-kirjaston funktioita. Kyseinen kirjasto mahdollistaa kuvien luonnin dynaamisesti sivuja generoitaessa. Esimerkiksi nelikulmiot saadaan piirrettyä käyttämällä imagerectangle()-funktiota, viivat imagelineimage()-funktiolla ja tekstit muodostetaan imageTTFText()-funktiolla, jonka parametrinä määritetään käytettävä TrueType-fontti. Listauksessa 13 on yksinkertainen esimerkki PNG-tyyppisen kuvan luomisesta. Imagecreate-funktiolle annetaan kuvalle haluttu koko parametreinä, jolloin paluuarvona saadaan $image-muuttujaan kahva, jota tämän jälkeen voidaan käyttää GD-kirjaston funktioiden parametreinä.

```
<?php
header("Content-type: image/png");
$image = imagecreate($width, $height);

...

imagepng($image);
imagedestroy($image);
?>
```

Listaus 13: Kuvan luominen PHP:n GD-kirjastolla

5 PALVELU KOKONAISUUDESSAAN

Keskeisessä osassa koko palvelussa ovat pelit, niiden käyttäjät ja heitä varten rakennetut erilaiset kommunikointijärjestelmät. Käyttäjien kokonaisuus muodostaakin työlle nimen yhteisöpalvelu. Sivusto sisältää monia ominaisuuksia, joita tässä työssä ei kuitenkaan syvällisesti käsitellä. Jäljempänä kerrotaan näistä oleellisimmat.

5.1 Pelitulosten ennätyslistat

Tällä hetkellä jokaisesta pelatusta yksinpelistä tallennetaan tulokset tietokantaan. Tämä on mahdollista hoitaa myös vain pääpalvelimen kautta, mikäli pelikohtaisella palvelimella ei tietokantaa olisi käytettävissä. Listauksessa 13 käytetään Javan Statement-luokkaa, jolla tallennetaan pelin tulos tietokantaan execute()-metodilla. Yksinpeleistä tallennettavat perustiedot ovat käyttäjän ID-tunnus, tulos ja päivämäärä.

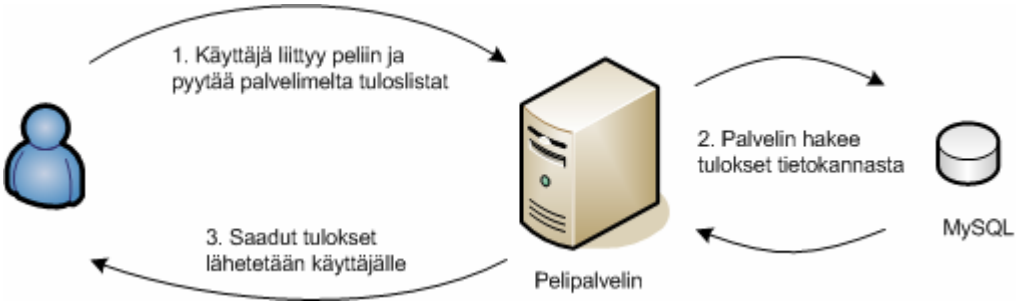
```
Statement stmt = connection.createStatement();

String query =
    "INSERT INTO game (id, user_id, date, result, mode) " +
    "VALUES (null, "+userID+", NOW(), "+result+", '"+mode+"')";
stmt.execute(query);
stmt.close();
```

Listaus 14: Yksinpelituloksen tallentaminen tietokantaan

Pelit ovat toteutettuina niin, että kun pelitulos lähetetään palvelimelle, ei uutta peliä ole mahdollista heti aloittaa. Vasta kun sovelma on vastaanottanut varmistuksen tuloksen tallentumisesta kuvan 15 mukaisesti, aktivoituu mahdollisuus uuteen peliin.

Kuva 15: Kun tulos on tallennettu tietokantaan, lähetetään varmistus käyttäjälle

Monissa yksinpeleissä ovat tulokset suoraan pelin aikana nähtävissä. Näissä tapauksissa aina käyttäjän liittyessä peliin, lähetetään palvelimelle tulostenpyyntöviesti, kuten kuvasta 16 ilmenee.



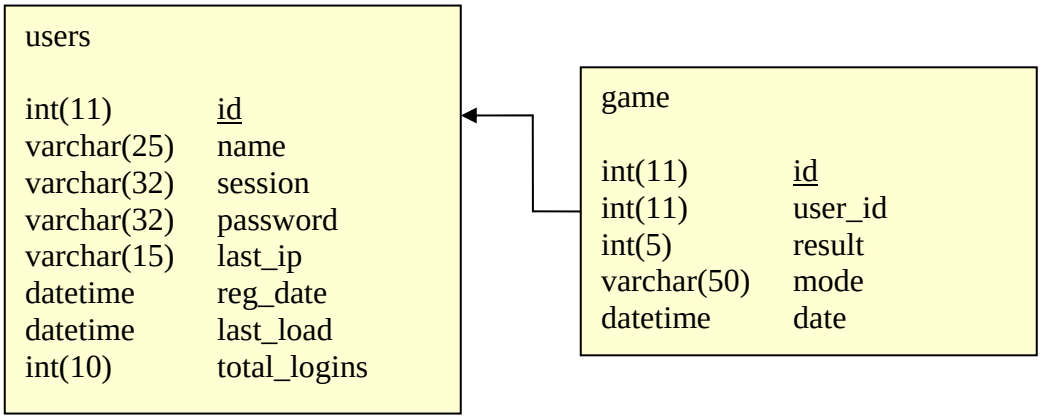
Kuva 16: Peliin liittyessään käyttäjä pyytää tuloslistat palvelimelta

Listauksessa 14 haetaan pelimoodin perusteella tuloslistat, joissa on kymmenen parasta tulosta. Kun tietokantakysely tuottaa hakutuloksia, viedään ne ResultSet-olioon. Tällä oliolla on next()-metodi, joka palauttaa aina tuloksista seuraavan. Kaikki tulokset voidaankin käydä läpi kyseisen metodin avulla while()-silmukalla esimerkin mukaisesti. Kaikki pelitulokset tallennetaan, jotta niistä esimerkiksi saadaan myöhemmin graafiset kuvaajat. Koska yhdelläkäyttäjällä on monta eri pelitulosta, käytetään tietokantakyselyssä max()-funktiota suurimman tuloksen hakemiseen. Tällä tavalla saadaan pelin ennätyslistat sellaiseen muotoon, että yksi nimi näkyy listalla enintään vain yhden kerran.

```
ResultSet rs = null;
Statement stmt = connection.createStatement();

String query =
    "SELECT users.nick, max(game.result) as result " +
    "FROM game, users" +
    "WHERE mode = '" + mode + "' AND users.id = game.user_id " +
    "GROUP BY game.user_id " +
    "ORDER BY result DESC LIMIT 10");

rs = stmt.executeQuery(query);
while(rs.next())
{
    ...
}
```

Listaus 15: Yksinpelitulosten hakeminen suoraan käyttäjän sovelmaan

Pelien ennätyslistoja varten keräämät tiedot tietokantataulussa (kuva 17) sisältävät käyttäjän ID-tunnuksen, pelituloksen ja päiväyksen. Näiden lisäksi peleillä saattaa olla lisäominaisuuksia, kuten esimerkiksi pelimoodi, joka pitää myös tallentaa. Taulussa user_id eli käyttäjän ID-tunnus tarkoittaa käyttäjätaulun pääavainta, josta löytyy käyttäjän tärkeimmät tiedot. Parhaimmat pelitulokset näkee usein suoraan peleistä käsin, mutta sivustolla on myös oma osio ennätyslistoille, joissa kaikki kerätyt tulokset näkyvät.



Kuva 17: Jokaisen pelin tuloksia varten on oma tietokantataulu

5.2 Ylläpidolliset hallintapaneelit

Kaikki oleelliset tapahtumat sivustolla ja peliaulassa tallentuvat lokitiedostoihin, jotta mahdolliset väärinkäytökset on mahdollista tarvittaessa selvittää. Väärinkäytöksistä yksittäisten käyttäjien pääsy palvelun tiettyihin osiin voidaan estää joko nimimerkin tai IP-osoitteen perusteella. Pienellä osalla käyttäjistä on erityisoikeudet, joilla pääsee sivuston hallintaosiin, joista muille käyttäjille on mahdollista antaa porttikiellot. Käyttäjien oikeustasot määräytyvät roolin perusteella, joita ovat ylläpidon jäsenet, foorumin moderaattorit, pelien valvojat ja normaalit käyttäjät. Valvojilla on oikeudet antaa porttikiellot palvelun käyttöehtoja rikkoville käyttäjille peleihin. Tämä onnistuu sekä peliaulan kautta että sivustolla olevan hallintasivun varsinaisella porttikieltotyökalulla. Moderaattoreilla on oikeus poistaa keskustelualueilta viestejä ja antaa porttikieltoja käyttäjille niin, että he eivät viestejä pääse lukemaan.

Ylläpitäjä pystyy lisäksi estämään käyttäjää pääsemästä koko sivustolle. Porttikieltojen tiedot ovat tietokannassa yhdessä taulussa, joissa yksi tietue sisältää alueen, jolle porttikielto vaikuttaa, porttikiellon saaneen ja sen antaneen käyttäjän ID-tunnukset ja päiväys, jolloin kielto lakkaa olemasta voimassa. Aina kun rekisteröitynyt käyttäjä käy sivustolla kirjautuneena, on hänen tärkeimmät tiedot tallennettuna PHP:ssä $user-taulukkoon. Tiedot sisältävät muun muassa käyttäjän ID-tunnuksen, jonka perusteella voidaan porttikiellot tarkistaa. Esimerkiksi keskustelualueiden porttikieltojen tarkistukseen ID:n perusteella käytetään PHP:llä toteutettua listauksen 14 mukaista tietokantakyselyä.

```
$sql = mysql_query("SELECT ban_reason
                    FROM bans
                    WHERE NOW() < ban_end_date
                    AND banned_id = $user_id
                    AND target_area = 'forum'")

if($sql && mysql_num_rows() > 0)
    return true;
else
    return false;
```

Listaus 16: Esimerkkikysely porttikiellon tarkistamisen käyttäjän ID:n perusteella

Rekisteröidyt käyttäjät ovat oleellisessa osassa koko palvelussa, joten näitä varten on tehty ylläpitoa helpottava kuva 18 mukainen hallintasivu. Käyttäjät ovat sivulla listattuna ja jokaisesta on tärkeimmät tiedot näkyvissä suoraan. Kunkin eri tiedon eli sarakkeen perusteella on mahdollista järjestää käyttäjät uudestaan.

	Nimimerkki	Kirjaut.	Rekisteröity	Viim. sivulataus	Ylläpit.	Moder.	Valvoja
muokkaa	rami	3919	08.12.2005 15:16	14.04.2007 18:12	Kyllä	Kyllä	Kyllä
muokkaa	admin	527	08.12.2005 15:17	25.01.2007 17:48	Ei	Ei	Ei
muokkaa	henri	443	09.12.2005 14:22	20.01.2007 17:40	Ei	Ei	Ei
muokkaa	Ylläpito	214	11.12.2005 17:12	29.01.2007 20:50	Ei	Ei	Ei
muokkaa	m	0	23.12.2005 13:46	09.06.2006 14:43	Ei	Ei	Ei
muokkaa	testi	330	23.12.2005 16:28	25.02.2007 12:01	Ei	Ei	Ei
muokkaa	devnull	141	03.01.2006 16:54	22.09.2006 12:05	Ei	Ei	Ei
muokkaa	ionic	4	03.01.2006 16:55	13.10.2006 01:26	Ei	Ei	Ei

Kuva 18: Osa ylläpidon käyttäjähallintasivusta, jossa ovat listattuna oleelliset tiedot kaikista käyttäjistä

Listassa jokaisen käyttäjän kohdalla on muokkauslinkki, jota klikkaamalla aukeaa kuvan 19 kaltainen lomake. Kyseisestä lomakkeesta saa näkyviin jokaiselta yksittäiseltä käyttäjältä hieman enemmän tietoja kuin edellä mainitusta käyttäjälistasta. Jokaiselle käyttäjälle tallennettuja perustietoja ovat yksilöllinen ID-tunnus, nimimerkki, viimeisen sivunlatauksen ajankohta, viimeisen kirjautumisen yhteydessä käytetty IP-osoite, kirjautumisten yhteismäärä, rekisteröitymispäivä, salasana kryptattuna ja erityiset oikeustasot.

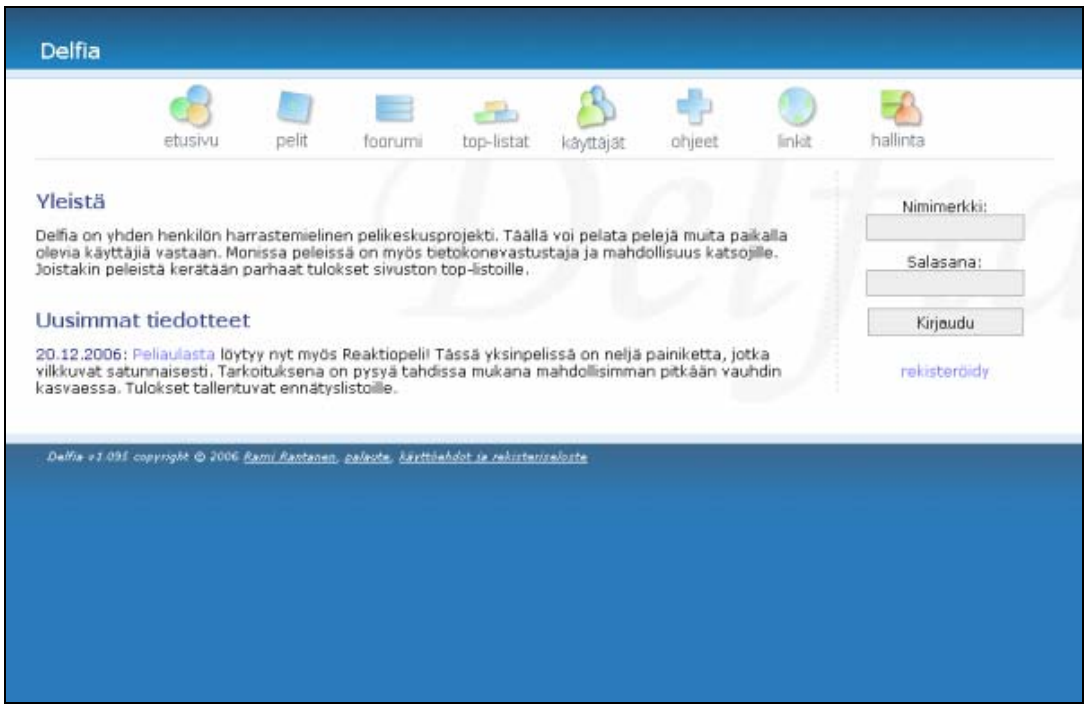
Lomakkeen ideana on ylläpidollisesti helpottaa käyttäjien hallintaa. Tällöin kaikkien oleellisten käyttäjätietojen muokkaus onnistuu helposti ja nopeasti. Lomakkeen kautta pystyy muokkaamaan nimimerkkiä, vaihtamaan salasanan ja määrittämään käyttäjän oikeustason. Myös pelitulosten nollaaminen ja käyttäjän poistaminen järjestelmästä onnistuu tämän kautta. Käyttäjän poistaminen tarkoittaa samalla myös kaikkien kyseiselle käyttäjälle viitattujen pelitulosten poistamista. Käyttäjien kontrollointi muiden hallintapaneelien ohella on saatavilla vain ylläpidon käyttöoikeuksilla ja vaatii lisäksi erillisen salasanan syöttämistä.

Kuva 19: Ylläpidon käytössä oleva yksittäisten käyttäjien tietojen muokkauslomake

5.3 Sivuston muut ominaisuudet

Yhteisöpalveluissa tärkeänä osana on käyttäjien välinen kommunikointi. Tällaisia ominaisuuksia on tähänkin palveluun edellisissä kappaleissa mainittujen asioiden lisäksi toteutettu monia. Suurimpana tällaisena ominaisuutena on kokonainen foorumijärjestelmä, joka sisältää täydelliset moderaattoritoiminnot viestien ja avausten käsittelyä varten. Näiden lisäksi käyttäjät voivat lähettää viestejä toisilleen yksityisesti. Jokaisella käyttäjällä myös oma vieraskirja, jonne muut voivat käydä kommentoimassa. Kuten kappaleessa 1. Johdanto mainittiin, näitä ei tässä työssä kuitenkaan käsitellä.

Käyttäjätunnusten rekisteröintimahdollisuus helpottaa yksilöllisten toimintojen toteuttamista koko palveluun ja tämä edellyttää Suomen henkilötietolain (523/1999) §10 mukaisesti, että sivustolla on kaikkien käyttäjien nähtävillä rekisteriseloste. Tästä selviää käyttötarkoitukset rekisteröitymisen yhteydessä kerätyille tiedoille. Kuvassa 20 on näkymä palvelun etusivusta. Tärkeimmät navigointilinkit ovat selkeästi yläreunassa, joiden kautta pääsee sivuston eri osiin.



Kuva 20: Näkymä palvelun etusivusta

6 YHTEENVETO

Palvelun rakentaminen alusta lähtien ja esiin tulleiden ongelmien ratkaiseminen oli erittäin opettavaista. Myöhemmässä vaiheessa yksityiskohtien parantaminen, vakauden varmistaminen ja koodirakenteiden optimointi oli haastavaa, mutta mielenkiintoista. Koko sivusto peliaulajärjestelmineen saavuttikin lopulta erittäin vakaan tilan ja tällä hetkellä palvelun saatavuus on hyvin lähellä 100 prosenttia. Varsinaisia ongelmia ei palvelun perusrakenteissa ole enää esiintynyt.

Tällä hetkellä käytössä oleva viestinvälitystekniikka asiakkaan ja palvelimen välillä on tehokas ja hyvin dynaamisesti kaikkiin tarpeisiin mukautuva oliovirta. Sitä olisi kuitenkin mahdollista kehittää tehokkaammaksi ja oliovirtojen sijasta voisi useassa eri tilanteessa käyttää pelkkää tekstimuotoista protokollaa. Tällöin pakettien siirtoviiveet olisivat oliopaketteja pienempiä ja palvelinsovellukset olisi mahdollista toteuttaa tehokkaammin matalan tason ohjelmointikielillä.

Tietoturvan osalta Javalla toteutettujen sovellusten voi ajatella olevan matalan tason kieliä turvallisempia, koska esimerkiksi puskurin ylivuotovirheitä ei pääse syntymään Javan sisäisen muistinhallinnan ansiosta. Sen sijaan yksi oleellinen tietoturvaseikka tällä hetkellä on se, että viestipaketit kulkevat asiakkaalta palvelimelle salaamattomina. Se tarkoittaa, että teoriassa on mahdollista kaapata peliaulan TCP-paketit ja esimerkiksi lähettää palvelimelle vääristettyjä pelituloksia. Varsinkin kriittisemmässä palvelussa tämä olisi syytä huomioida.

Erityisinä tavoitteina tässä työssä oli saada rakennettua monipuolinen järjestelmä, jossa kokonaisuus pysyy selkeänä ja täysin hallinnassa. Asiat pyrittiin toteuttamaan ylläpidollisesti helpoiksi, ja tarkoituksena oli erityisesti huomioida vakaus, tehokkuus ja käytettävyys. Nämä asetetut tavoitteet saatiin toteutettua hyvin ja ajallaan. Palvelu on jo tähän mennessä saanut paljon positiivista palautetta ja sen suosio on lisääntynyt hyvää vauhtia.

LÄHDELUETTELO

1. Java 2 Standard Edition API. [www-sivu]. [viitattu 1.3.2007] Saatavissa: http://java.sun.com/j2se/1.4.2/docs/api/
2. PHP Manual. [www-sivu]. [viitattu 1.3.2007] Saatavissa: http://fi.php.net/manual/en/
3. MySQL Manual. [www-sivu]. [viitattu 1.3.2007] Saatavissa: http://dev.mysql.com/doc/refman/5.0/en/index.html
4. Java Optimization. [www-sivu]. [viitattu 1.3.2007] Saatavissa: http://www.glenmccl.com/jperf/
5. Haikala, Ilkka, Ohjelmistotuotanto, 8. painos. Talentum Media. Pieksämäki 2002. s. 35 – 41, 78 – 83.
6. Harsu, Maarit, Ohjelmien ylläpito ja uudistaminen. Gummerus. Jyväskylä 2003. s. 86 – 88.